行動應用 App 安全開發指引(草案)

經濟部工業局
中華民國105年10月

目　次

圖 目 次

表 目 次

1. 前言

1.1. 目的

配合經濟部工業局「行動應用 App 基本資安規範」，設計 Android 與 iOS 作業系統之「行動應用 App 安全開發指引」(以下簡稱本指引)，以培養行動應用 App(以下簡稱行動 App 或 App)開發人員具備編寫安全程式碼之技能。

由於行動智慧裝置具備高度移動性、整合雲端服務、幾近全時貼近持有使用者之特性，App 開發人員在功能優先導向前提下，常會忽略資訊安全之管控措施，以不安全程式開發實務或引用不安全第三方函式庫，造成安全性漏洞，導致惡意攻擊者可藉著取得行動智慧裝置的存取權限，或透過網路進行中間人攻擊等行為，進而竊取傳遞中敏感性資料或是在使用者未授權狀況下調用付費資源。

「安全是設計出來，不是被駭出來」，軟體安全是軟體品質中最重要一環，因此本指引旨在檢視當前行動應用 App 之資安威脅與風險，開發人員應在開發生命週期需求階段，將使用者隱私保護及資料安全納入安全基本需求，不待業主或需求單位提出安全需求，即應將安全設計於 App 或系統架構中。

考量行動應用 App 開發專案週期短、更新快且開發人員多數屬於小型工作室或個人，如何將安全軟體發展生命週期(Secure Software Development Life Cycle,SSDLC)流程簡化，導入類似敏捷式開發，並歸納各國行動應用 App 開發實務及程式碼範例供開發人員選用，設計及開發出符合經濟部工業局「行動應用 App 基本資安檢測」要求五大面向，並鼓勵自主通過 TAF 認證第三方行動應用 App 安全檢測實驗室檢測，取得 MAS 標章，避免因開發疏忽產出具資安

漏洞及可防範惡意攻擊之 App 安全程式，以保護使用者敏感性資料。

1.2.適用對象

1.2.1. 中大型組織或開發業者

行動應用 App 開發相關人員：專案經理(Project Manager, PM)、系統分析人員(System Analyst, SA)、資訊安全人員(Information Security, IS)、系統設計人員(System Developer, SD)、程式設計師(Programmer, PG)、軟體檢測人員(Quality Assurance, QA)、資料庫管理員(Data Base Administrator, DBA)、資訊維運人員(Management Information System, MIS)。

1.2.2. 小型工作室

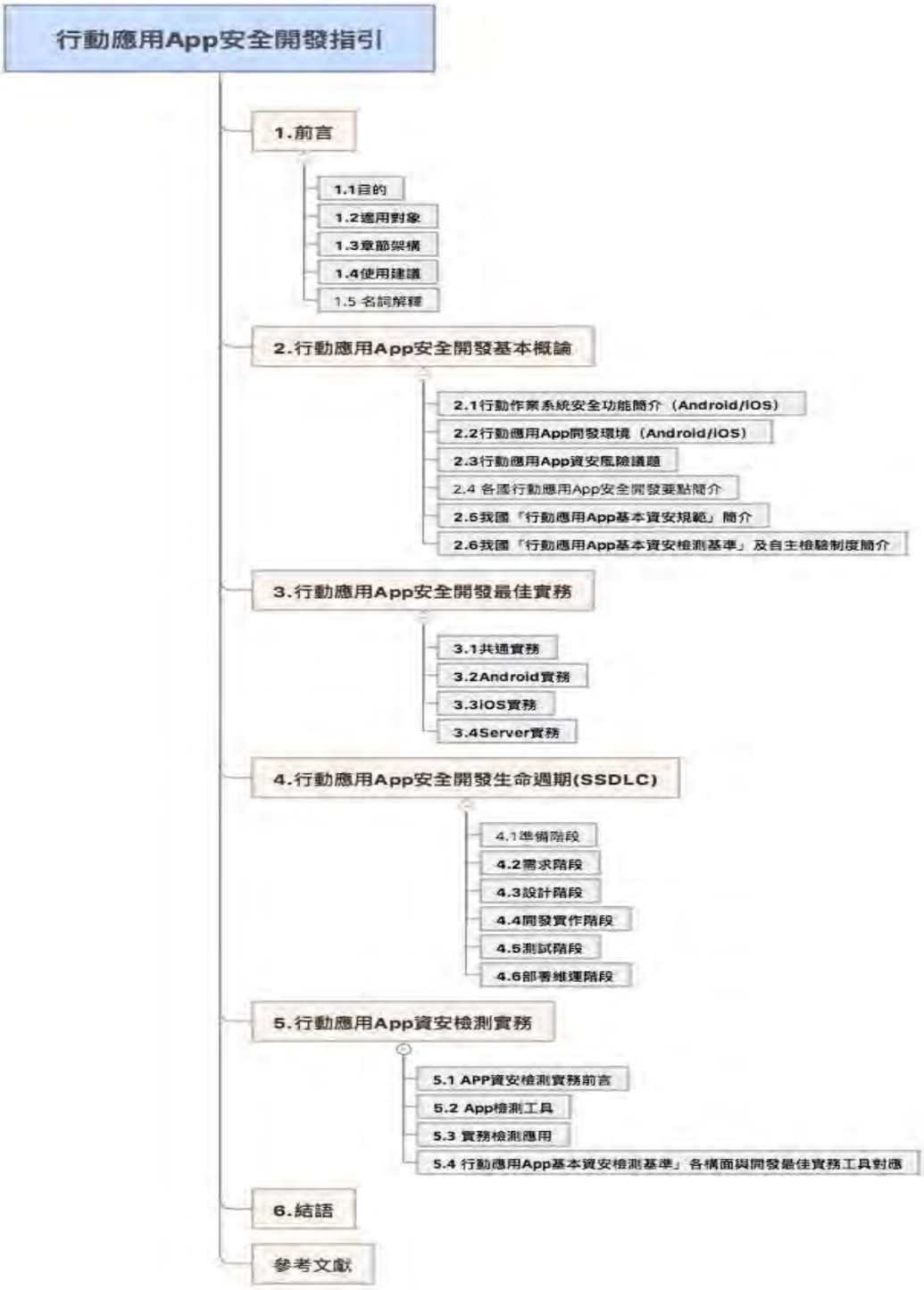
系統分析人員、程式設計人員。

1.2.3. 個人開發人員

個人開發人員。

1.3.章節架構

本指引章節架構，詳見圖 1 所示。



資料來源：本計畫整理

圖1　指引章節架構圖

1.3.1. 第 1 章前言

本章說明本指引目的、適用對象、章節架構、使用建議及名詞解釋等，使讀者易於了解本指引全貌，視需求選取適用及需要的內容。

1.3.2. 第 2 章行動應用 App 安全開發概論

本章區分為「行動作業系統安全功能簡介(Android/iOS)」、「行動應用 App 開發環境(Android/iOS)」、「行動應用 App 資安風險議題」、「各國行動應用 App 安全開發要點簡介」、「我國行動應用 App 基本資安規範簡介」及「我國行動應用 App 基本資安檢測基準及自主檢驗制度簡介」等 6 節，目的在於使讀者在進入 App 開發專案流程前，能對相關安全議題有一定之知識基礎，同時透過不同之行動作業平台、開發環境、安全組態及開發模式介紹，使讀者能了解軟體安全並非處於靜止狀態，需要因應環境變動不斷尋求合適解決方案。

- 第 2.1 節「行動作業系統安全功能簡介(Android/iOS)」

  簡述 Android 與 iOS 作業系統架構、內建安全功能，例如：檔案系統架構、App 運行環境及加解密系統，以使讀者了解在行動應用 App 開發時可運用之原生安全功能。

- 第 2.2 節「行動應用 App 開發環境(Android/iOS)」

  以目前最流行之 Android 與 iOS 作業系統，說明 Java、Object-C 及 Swift 之官方提供整合性安全開發環境(Xcode、Android Studio)、提供安全開發程式設計指引及 API 等。

- 第 2.3 節「行動應用 App 資安風險議題」

說明行動應用 App 常見之資安風險，包括：OWASP 針對瀏覽器、作業系統、App 軟體、網路及伺服器端的攻擊，使讀者了解行動應用 App 攻擊面，以利於在需求、設計階段進行風險分析及開發實作時可以了解最佳實務安全目標。

- 第 2.4 節「各國行動應用 App 安全開發要點簡介」

  介紹國內外不同政府機關及團體組織對於行動智慧裝置之安全規範及標準，如美國國家標準機構(National Institute of Standards and Technology,NIST)、歐盟網路與資訊安全局(The European Network and Information Security Agency,ENISA)、日本智慧型手機安全協會(Japan Smartphone Security Forum,JSSEC)、中國大陸工業信息部、開放網頁 App 安全計畫(The Open Web Application Security Project, OWASP)及雲端安全聯盟(Cloud Security Alliance, CSA)等，使讀者未來於開發行動應用 App 時，可了解行動智慧裝置重要基本資安要求，並透過不同國家、組織發布之相關訊息，掌握最新安全風險動態與標準，以作為安全開發基礎。

- 第 2.5 節「我國「行動應用 App 基本資安規範」簡介」

  說明我國對於行動應用 App 之基本資安規範，包括對於程式發布、敏感資料、付費資源、身分認證與授權、網路連線及程式碼基本之安全技術要求，開發人員可參考此規範作為軟體開發之初步安全規劃，後續並透過安全軟體生命週期、安全檢測等手段提升行動應用 App 之安全品質。

- 第 2.6 節「我國「行動應用 App 基本資安檢測基準」及自主檢驗制度簡介」

本節將說明安全開發實務與我國行動應用 App 基本資安檢測基準之對應關係，俾利未來面對相關安全問題時，能靈活運用各項檢測基準、工具及方法。

另簡介「行動應用 App 基本資安自主檢測推動制度」，說明其制度規章、實驗室資格認證及管理規範及行動應用 App 基本資安標章(MAS)使用與管理規範。

1.3.3. 第 3 章安全行動應用 App 開發最佳實務

本章將說明行動應用 App 在開發實務上須注意之安全開發事項，並輔以不安全與安全程式碼範例，供讀者於開發作業時參考運用。

區分為共通安全開發實務準則、Android 安全開發實務、iOS 安全開發實務及伺服器安全實務等 4 小節說明，提供讀者了解安全開發最佳實務因應安全問題、解決方法，並輔以程式碼範例，使未來能實際運用於行動應用 App 開發作業。

●3.1 共通安全開發實務準則

本節整理 Android 及 iOS 有關「行動 App 發布安全」、「敏感性資料保護」、「付費資源控管安全」及「行動 App 碼安全」等共通安全開發實務計 102 項。

●Android 安全開發實務

包括檔案權限安全、保護程式服務及避免意圖(intent)監聽等 10 項。

●iOS 安全開發實務

包括 iCloud 鑰匙圈安全、避免程式、快取資料及快照備份等 4 項。

●伺服器安全開發實務

包括適當的網頁伺服器設定、良好的伺服器端 SSL 設定、使用適當的會話(session)管理等 12 項。

1.3.4. 第 4 章行動應用 App 安全開發生命週期

針對安全軟體發展生命週期(SSDLC)內容區分為「準備階段」、「需求階段」、「設計階段」、「開發實作階段」、「測試階段」及「部署維運階段」說明如下：

●第 3.1 節「準備階段」

本階段需要先建立行動應用 App 開發人員相關資訊安全意識，建立安全開發流程及接受行動應用 App 安全開發實務教育訓練。

●第 3.2 節「需求階段」

本階段應由 PM 及 SA 人員針對 App 軟體整體安全需求進行評估與分析，包括：識別 Android 與 iOS 作業系統及軟體可能面臨之安全風險，並完成安全與隱私風險評鑑，以了解各種安全威脅與隱私風險狀況；建立並定義正常與錯誤使用(濫用)案例，以利將業主或需求單位未於訪談時表達的潛在需求具體化，並對後續開發實作、測試對應需求提供參考模型。

●第 3.3 節「設計階段」

安全需求與隱私防護功能應及早於系統設計初期納入，以避免於開發後期再行納入，導致大幅增加成本。SD 人員實際執行時應針對相關功能架構描述安全實作方法，如有必要須與 DBA 與伺服器維運人員(MIS)討論架構設計可行性與安全性，包括：威脅建模、限制

非必要服務、最小權限法則及強化縱深防禦等採取手段，可依據行動應用 App 基本資安檢測基準建立查核點。

- 第 3.4 節「開發實作階段」

  包括：建立經過認可之安全開發工具清單，對於所有使用之函式庫應審查其安全歷史，並研擬安全之替代方案，程式開發人員(PG)須依行動應用 App 安全開發實務撰寫安全程式碼並實施單元測試，可依據行動應用 App 基本資安檢測基準建立查核點。

  同時本階段應執行程式碼靜態分析，可使用靜態自動分析工具或由主管或不同開發人員實施必要之人工審核。

  於本階段介紹敏捷式開發及 DevOps(Development 和 Operations 的組合詞)之持續整合、持續交付及持續部署，以加速 App 產品化流程。

- 第 3.5 節「測試階段」

  為確保行動應用 App 安全功能如預期般正常運作，除前期之靜態分析外，並應針對程式進行執行期動態測試，如 QA 人員使用工具檢視軟體運作狀況，包括：記憶體使用、使用者權限及其他重要安全問題。

  另應透過輸入不正確或亂數資料進行模糊測試，甚至採取滲透測試，以利及早發掘程式安全問題，並藉此修正與評估原有威脅建模疏漏之處，尤其應善加利用行動應用 App 基本資安檢測基準建立查核點，有關安全基本檢測實務部分將於後續章節做詳細說明。

- 第 3.6 節「部署維運階段」

行動應用 App 進入部署維運階段時，MIS 人員應先對作業系統、伺服器、資料庫及軟體本身等部署環境進行安全檢視工作，包括：使用正確的安全設定、關閉非必要之服務與網路埠、移除測試帳號與資料，以及使用最低權限執行等。

遇有軟體或作業系統環境變動時，應做好變動管理工作，包括：識別可能變動及分析變動對安全性之影響，另應進行持續性之監測，以驗證軟體運作安全性。

1.3.5. 第 5 章行動應用 App 安全檢測實務

本章建議開發人員發展之行動應用 App 安全檢測活動，應由開發人員以外獨立測試人員或第三方單位人員進行檢測，透過靜、動態分析、黑箱及白箱等測試方式，可針對行動應用 App 的安全性議題分別進行探討，有助於排除開發人員自我開發之盲點。

此外，獨立測試人員可由測試工程方法蒐集與探討相關惡意或疑慮之函式庫、線上服務介面或相關第三方程式模組等，並且研擬相關測試方案、表單、報告及改善方法，有助於團隊之分工合作以及行動應用 App 提升改版之效率與品質。

1.3.6. 第 6 章結語

說明本指引之預期成果及其他未來發展注意事項，讀者除藉本指引強化行動應用 App 安全性外，並應隨時追蹤注意外在安全環境之變化，包括：軟體、工具弱點、漏洞及新型安全威脅之產生，以有效達成軟體安全開發之目標。

1.4.使用建議

本指引雖以 App 開發人員為主要閱讀對象，但 App 開發專案實務上會因 App 的需求目的、目標使用者、專案時程、及整合其他服務等

因素，必須與需求提案人員、專案經理、系統分析、系統設計、測試/品管及資訊安全人員等不同角色進行協同合作。

本指引為兼顧上述人員對安全開發需求，除了第 1 章「前言」為本指引閱讀地圖，第 6 章「結語」為指引總結，為各角色詳讀章節外，且第 3 章「行動應用 App 安全開發最佳實務」為核心，可以作為各種不同角色溝通共通語言。茲就針對不同角色，提供本指引各章節之閱讀建議如下：

1.4.1. 開發人員(PG)

- 在第 2 章「行動應用 App 安全開發概論」，開發人員即使對第 2.2 節「行動應用 App 開發環境(Android/iOS)」很熟悉，但不一定熟知 Android/iOS 作業平台內建可運用的安全機制如何運用，建議詳讀第 2.1 節「行動作業系統及安全功能簡介」。
  - 多數 App 開發人員並未具備資訊安全風險意識，可能因為不安全的行動應用 App 開發實務造成漏洞及可能利用這些的漏洞的威脅有那些，可以使開發人員認知過去未曾注意到安全的盲點，建議詳讀 2.3 節「行動應用 App 資安風險議題」。
  - 第 2.4 節「各國行動應用 App 安全開發要點簡介」、第 2.5 節我國「行動應用 App 基本資安規範」簡介、第 2.6 節「我國「行動應用 App 基本資安檢測基準」及自主檢驗制度簡介」，此 3 節相關安全開發實務已整理歸納在第 3 章「行動應用 App 安全開發最佳實務」中，建議 PG 人員可視需求選擇閱讀。
- 第 3 章「行動應用 App 安全開發最佳實務」，對於 App 開發人員建議詳讀如下：

– 第 3.1 節「共通安全開發實務準則」，可依「行動應用 App 基本資安規範」及「行動應用 App 基本資安檢測基準」五大項分類：「行動應用 App 發布安全」、「敏感性資料保護」、「付費資源控管安全」、「行動 App 使用者身分認證、授權與連線管理安全」與「行動應用 App 程式碼安全」之 17 項次分類去參考適用實務。

– 第 3.2 節「Android 安全開發實務」為 Android 版本 App 開發人員應詳讀、第 3.3 節「iOS 安全開發實務」為 iOS 版本 App 開發人員應詳讀、第 3.4 節「伺服器安全實務」為負責後端服務開發人員應詳讀，對於 Android 與 iOS App 開發人員可視需求選擇閱讀。

- 第 4 章「行動應用 App 安全開發生命週期(SSDLC)」，與開發人員直接相關為「開發實作階段」需詳讀，其餘「準備階段」、「需求階段」、「設計階段」、「測試階段」及「部署維運階段」可視需求選擇閱讀。
- 第 5 章「行動應用 App 安全檢測實務」主要是為測試人員或是第三方人員的測試相關實務，可視需求選擇閱讀。

1.4.2. 專案經理(PM)

PM 在本指引定義為專案負責人或為需求單位與客戶連絡窗口，閱讀建議如下，

- 第 2 章需要了解及熟悉第 2.1 節「行動作業系統及安全功能簡介(Android/iOS)」、第 2.2 節「行動應用 App 開發環境(Android/iOS)」、第 2.3 節「行動應用 App 資安風險議題」、第 2.5 節「我國「行動應用 App 基本資安規範」簡

介」、第 2.6 節「我國「行動應用 App 基本資安檢測基準」及自主檢驗制度簡介」、須視 App 發布國家需要選擇閱讀第 2.4 節「各國行動應用 App 安全開發要點簡介」，以利進行專案安全需求規劃，建議詳讀第 2 章。

- 第 3 章行動應用 App 安全開發最佳實務，主要依照 App 的安全分類，建議重點閱讀各安全開發實務準則項目的主題與說明內容即可，其中程式碼範例主要是提供 SA/SD/PG 人員研讀。
- 第 4 章「行動應用 App 安全開發生命週期(SSDLC)」部分跟整體 App 開發專案息息相關，包括：使用 SSDLC 各階段安全活動是否被執行、時程規劃與品質及進度監控等，以利與 SSDLC 各階段不同角色溝通及協調，建議詳讀第 4 章。
- 第 5 章「行動應用 App 安全檢測實務」主要是提供給測試人員或是第三方人員的測試相關實務，建議進行重點閱讀。

1.4.3. 系統分析(SA)/系統設計(SD)

- 第 2 章「行動應用 App 安全開發概論」閱讀建議如下：
  - SA/SD 人員對 Android/iOS 作業平台內建可運用的安全機制均需熟悉，建議詳讀第 2.1 節「行動作業系統及安全功能簡介」及第 2.2 節「行動應用 App 開發環境簡介(Android/iOS)」。
  - 需求單位及客戶可能只會提出功能性需求，安全性需求可能需要由 SA 人員視 App 的功能、蒐集、處理、利用及傳送資料類別及使用環境定義安全需求及使用案例，交由 SD 或 PG 進行設計或實作，建議閱讀 2.3 節「行動應用 App 資安風險議題」以了解行動應用 App 可能因為不安全的開發實務造成漏洞及可能利用這些漏洞的威脅，將控制 App 安全風險措施納入系統規格文件。

- 第 3 章「行動應用 App 安全開發最佳實務」閱讀建議如下：
  - 第 3.1 節共通安全開發實務準則，可依「行動應用 App 基本資安規範」及「行動應用 App 基本資安檢測基準」五大項分類：「行動應用 App 發布安全」、「敏感性資料保護」、「付費資源控管安全」、「行動 App 使用者身分認證、授權與連線管理安全」與「行動應用 App 程式碼安全」之 17 項次分類去參考適用實務。
  - 第 3.2 節為 Android 專有安全開發實務、3.3 節為 iOS 專有安全開發實務，3.4 節為伺服器安全實務，SA/SD 人員原則上可全面瀏覽，再依 App 專案安全性依需要選擇並詳細閱讀，以利規劃、設計 App 安全控管機制。
- 第 4 章「行動應用 App 安全開發生命週期(SSDLC)」，與 SA/SD 人員直接相關為「需求階段」與「設計階段」需詳讀，其餘「準備階段」、「開發實作階段」、「測試階段」及「部署維運階段」可視需求選擇閱讀。
- 第 5 章「行動應用 App 安全檢測實務」主要是為測試人員或是第三方人員的測試相關實務，SA/SD 人員可視需求選擇閱讀。

1.4.4. 測試/品管人員(TEST/QA)

TEST/QA 人員，主要專注於 App 整合性測試與軟體品質，提供閱讀建議如下：

- 第 2 章「行動應用 App 安全開發概論」，除第 2.6 節「我國「行動應用 App 基本資安檢測基準」及自主檢驗制度簡介」建議詳讀，其餘小節可視需求選擇閱讀。

- 第 3 章「行動應用 App 安全開發最佳實務」，除建議重點閱讀 SSDLC 之「測試階段」適用的各項實務準則，其餘可視需求選擇閱讀。
- 第 4 章「行動應用 App 安全開發生命週期(SSDLC)」與 TEST/QA 人員直接相關為第 4.5 節「測試階段」需詳讀，其餘「準備階段」、「需求階段」、「設計階段」、「開發實作階段」及「部署維運階段」可視需求選擇閱讀。
- 第 5 章「行動應用 App 安全檢測實務」主要為測試人員或是第三方人員的測試相關實務，建議 TEST/QA 人員詳讀。

1.4.5. 資訊系統管理員(MIS)

MIS 人員因負責 App 產品上線後之系統維運，提供閱讀建議如下：

- 第 2 章「行動應用 App 安全開發概論」，可視需求選擇閱讀。
- 第 3 章「行動應用 App 安全開發最佳實務」與 MIS 人員直接相關為 SSDLC 之「部署維運階段」，建議詳讀風 3.4 節「伺服器安全實務」，其餘建議重點閱讀適用「部署維運階段」之各項實務準則。
- 第 4 章「行動應用 App 安全開發生命週期(SSDLC)」與 MIS 人員直接相關為第 4.6 節「部署維運階段」需詳讀，其餘「準備階段」、「需求階段」、「設計階段」、「開發實作階段」及「測試階段」可視需求選擇閱讀。
- 第 5 章「行動應用 App 安全檢測實務」，可視需求選擇閱讀。

1.4.6. 資安人員(IS)

IS 人員因需要確保整個 SSDLC 過程所有安全活動均被有效落實，建議全面瀏覽本指引，但不需要如各角色對其相關部分詳讀，具備基本概念即可。

其中第 2.3 節「行動應用 App 資安風險議題」、2.5 節我國「行動應用 App 基本資安規範」簡介、第 2.6 節我國「行動應用 App 基本資安檢測基準」與自主檢驗制度簡介及第 4 章「行動應用 App 安全開發生命週期(SSDLC)」為行動 App 安全活動重點建議詳讀，可將本指引視為 App 開發專案安全協調共通語言。

綜整開發生命週期、閱讀對象及本指引章節，製作本指引閱讀建議對照表，詳見表 1 所示。

表1 指引使用建議表

角色 / 章節	PM	SA	SD	Android PG	iOS PG	TEST/ QA	MIS	IS
SSDLC	全程	需求	設計	開發實作	開發實作	測試	部署維運	全程
Ch.1	●	●	●	●	●	●	●	●
2.1	●	●	●	●	●	○	○	●
2.2	●	●	●	○	○	○	○	◐
2.3	●	●	●	●	●	○	○	●
2.4	●	○	○	○	○	○	○	◐
2.5	●	○	○	○	○	○	○	●
2.6	●	○	○	○	○	●	○	●

角色 / 章節	PM	SA	SD	Android PG	iOS PG	TEST/ QA	MIS	IS
3.1	◐	◐	◐	●	●	◐	◐	◐
3.2	◐	◐	●	●	○	◐	◐	◐
3.3	◐	◐	●	○	●	◐	◐	◐
3.4	◐	◐	●	○	○	◐	●	◐
4.1	●	●	●	●	●	○	●	●
4.2	●	●	○	○	○	○	○	●
4.3	●	○	●	○	○	○	○	●
4.4	●	○	○	●	●	○	○	●
4.5	●	○	○	○	○	●	○	●
4.6	●	○	○	○	○	○	●	●
5.1	◐	○	○	○	○	●	○	○
5.2	◐	○	○	○	○	●	○	○
5.3	◐	○	○	○	○	●	○	○
5.4	◐	○	○	○	○	●	○	○
Ch.6	●	●	●	●	●	●	●	●

圖例：●：詳讀 ◐：重點閱讀 ○：可視需求選擇

資料來源：本計畫整理

1.5.名詞解釋

1.5.1. 行動應用 App(Mobile Application)

指一種設計給智慧型手機、平板電腦和其他行動裝置使用之 App。

1.5.2. 行動 App 商店(Application Store)

指行動裝置使用者透過內建在裝置中之行動 App 商店或透過網站對應用程式、音樂、雜誌、書籍、電影、電視節目進行瀏覽、下載或購買。

1.5.3. 敏感性資料(Sensitive Data)

指依使用者行為或行動 App 之運作，建立或儲存於行動裝置及其附屬 儲存媒介之資訊，而該資訊之洩漏有對使用者造成損害之虞，包括但不限於個人資料、通行碼、即時通訊訊息、筆記、備忘錄、通訊錄、地理位置、行事曆、通話紀錄及簡訊。

1.5.4. 個人資料(Personal Data)

指主要依「個人資料保護法」第二條上定義之所有得以直接或間接方式識別個人之資料，包括自然人之姓名、出生年月日、國民身分證統一編號、護照號碼、特徵、指紋、婚姻、家庭、教育、職業、病歷、醫療、基因、性生活、健康檢查、犯罪前科、聯絡方式、財務情況、社會活動及其他得以直接或間接方式識別該個人之資料。

國際行動設備識別碼(International Mobile Equipment Identity, IMEI)、國際行動用戶識別碼(International Mobile Subscriber Identity, IMSI)雖不在我國個人資料保護法第二條規範個人資料範圍內，但在行動智慧裝置屬於唯一識別碼，可間接識別個人的資料，本指引將 IMEI/IMSI 納入個人資料範圍。

1.5.5. 通行碼(Password)

指能讓使用者完全或有限度之使用系統或取得一組資料之識別使用者身分用之字元串。

1.5.6. 付費資源(In-App Purchase/Billing)

指透過行動 App 內建購買功能取得之額外功能、內容及訂閱項目。

1.5.7. 交談識別碼(Session Identification, Session ID)

指在建立連線時，指派給該連線之識別碼，並做為連線期間之唯一識別碼；當連線結束時，該識別碼可釋出並重新指派給新之連線。

1.5.8. 憑證(Certificate)

指載有簽章驗證資料，提供終端與伺服器間身分鑑別及資料傳輸加密。

1.5.9. 憑證機構(Certificate Authority)

指簽發憑證之機關、法人。

1.5.10. 惡意程式碼(Malicious Code)

指在未經使用者同意之情況下，侵害使用者權益，包括但不限於任何具有惡意特徵或行為之程式碼。

1.5.11. 資訊安全漏洞(Vulnerability)

指行動 App 安全方面之缺陷，使得系統或行動 App 資料之保密性、完整性及可用性面臨威脅。

1.5.12. 函式庫(Library)

指將一些繁複或者牽涉到硬體層面之程式，包裝為函式(Function)或物件(Object)蒐集在一起，編譯成二進位碼提供程式設計者使用。

1.5.13. 注入攻擊(Code Injection)

指因行動 App 設計缺陷而執行使用者所輸入之惡意指令，包括但不限於命令注入(Command Injection)、資料隱碼攻擊(SQL Injection)。

1.5.14. 行動智慧裝置(Smart Device)

指一種電子裝置具備攝影機、錄音機、GPS 及動作感應器，可安裝作業系統經由不同無線網路協定如藍牙、NFC、Wi-Fi、CDMA 或其他無線模組及可空中下載軟體，可獨立操作連結資料中心或雲端服務，這類設備可能是智慧型手機、平板電腦、穿戴式裝置及智慧家電等。

1.5.15. ARM 架構

ARM 架構，過去稱作進階精簡指令集機器(英語：Advanced RISC Machine，更早稱作：Acorn RISC Machine)，是一個 32 位元精簡指令集(RISC)處理器架構，其廣泛地使用在許多嵌入式系統設計。但在其他領域上也有很多作為，由於節能的特點，ARM 處理器非常適用於行動通訊領域，符合其主要設計目標為低成本、高效能及低耗電的特性。另外一方面，超級電腦消耗大量電能，ARM 同樣被視作更高效的選擇。

1.5.16. App 沙箱(Application Sandbox)

在資訊安全領域，沙箱(英語：sandbox，又譯為沙盒)是一種安全機制，為執行中的程式提供的隔離環境。通常是作為一些來源不可信、具破壞力或無法判定程式意圖的程式提供實驗之用。

沙箱通常嚴格控制其中的程式所能存取的資源，如：沙箱可以提供用後即回收的磁碟及記憶體空間。在沙箱中，網路存取、對真實系統的存取、對輸入裝置的讀取通常被禁止或是嚴格限制。從這個角度來說，沙箱屬於虛擬化的一種。

1.5.17. 空中下載(OTA)

透過無線傳輸方式，進行軟體、參數及相關資料之下載或更新。

2. 行動應用 App 安全開發基本概論

2.1.行動作業系統及安全功能簡介(Android/iOS)

2.1.1. Android 作業系統安全機制

2.1.1.1. Android 安全概觀

2.1.1.1.1. Android 屬於行動智慧裝置開放平台

透過開放可以讓行動智慧裝置製造商及開發商運用，相較封閉平台開放更多創新空間及運用先進軟硬體環境，帶給使用者良好的使用者體驗，但同時仍需要兼顧平台的安全性，保護使用者敏感性資料不易被惡意竊取。

2.1.1.1.2. 安全架構和安全程序

為了保護一個開放的平台，需要一個強大的安全架構和嚴格的安全程序，Google Android 開發團隊設計採用了多層的安全性，提供了一個開放的平台所需的靈活性及對開發人員設計 App 提供保護。有關安全問題回報和更新過程的資訊，請參閱安全更新和資源（查詢時間：2016/10/1)。

2.1.1.1.3. Android 的安全設計考慮到開發商能力成熟度

平台提供安全控制考量係以減少開發人員的負擔而設計。有安全意識且具備豐富經驗的開發人員能夠輕鬆使用和依賴靈活的安全控制。不太熟悉自訂安全控制開發人員，則可以透過使用平台的預設安全設定得到一定程度保護。

2.1.1.1.4. Android 的安全設計充分考量設備使用者面對安全風險

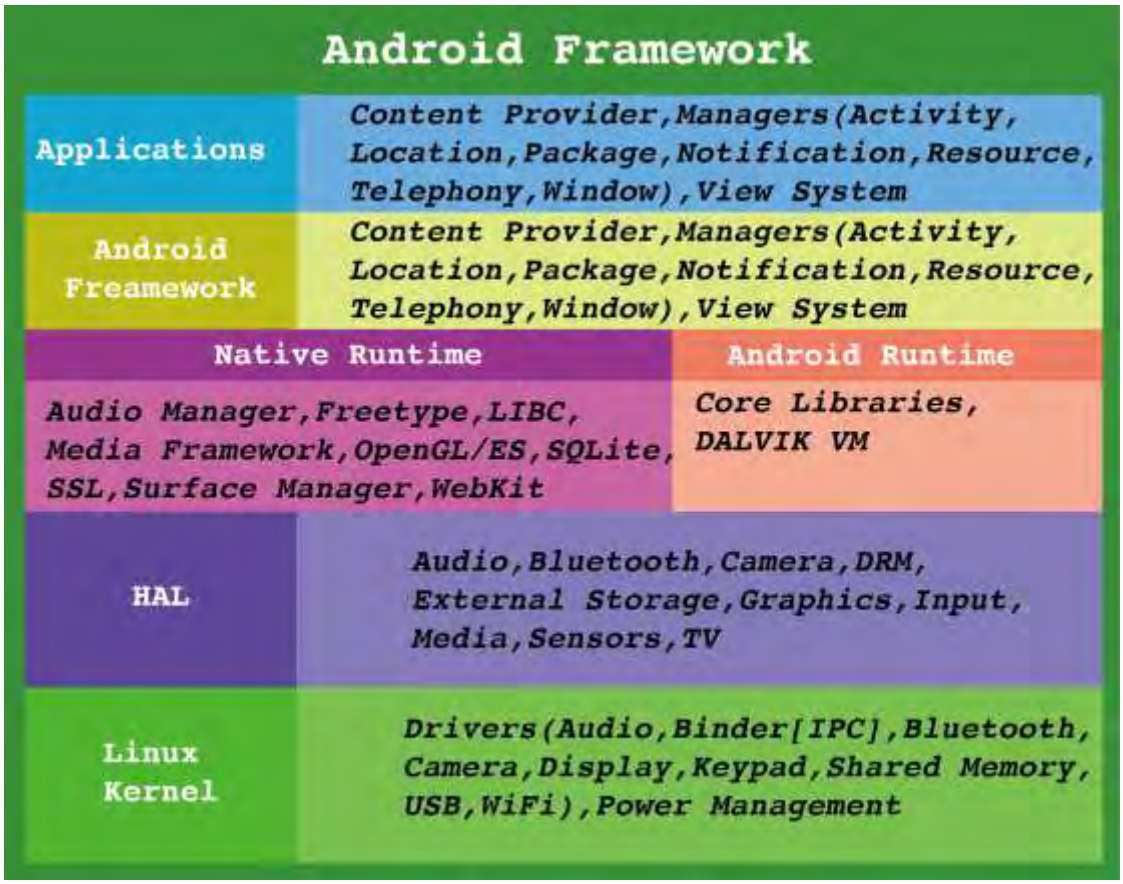
Android 的安全設計考量包括攻擊者會試圖執行常見的攻擊類型，如社交工程攻擊來騙取設備使用者安裝惡意軟體和 Android 上的第

三方 App 的攻擊的可能性。Android 在設計上，除減少這些攻擊的可能性，並大大限制了被攻擊成功事件的影響程度。為此 Android 建立了安全軟體架構與持續性的安全維護計畫(詳見 2.1.1.3 安全維護計畫)。

2.1.1.2. Android 架構安全設計

總結各級 Android 軟體堆疊的安全元件和注意事項詳見圖 2 所示。

每個元件假設其下層元件適當被保護。隨著以 root 身分運行 Android 作業系統小量的程式碼之外，Linux 內核以上的所有程式碼是由 App 沙箱的限制存取。



資料來源：Android Security Overview（查詢時間：2016/10/1)

圖2　Android 軟體堆疊架構

2.1.1.2.1. 主要的 Android 平台底層

●設備硬體

運行 Android 的廣泛的硬體裝置，包括智慧型手機，平板電腦和機上盒。Android 是與處理器無關的作業平台，但有時確實需要一些硬體的特定優勢安全功能，如 ARM v6 的執行-決不(eXecute-Never)決定該頁是否可以被執行權限。

●Android 作業系統

Android 的核心作業系統是採用 Linux 內核。App 存取設備內所有資源，如照相功能、全球定位系統、藍牙功能、電話功能及網路連結等皆透過作業系統存取。

●Android 運行階段

Android Runtime(縮寫為 ART)，是一種在 Android 作業系統上的運行環境，由 Google 公司研發，並在 2013 年作為 Android 4.4 系統中的一項測試功能正式對外發布，在 Android 5.0 及後續 Android 版本中作為正式的運行時，函式庫取代了以往的 Dalvik 虛擬機。

ART 能夠把應用程式的位元組碼(bytecode)轉換為機器碼，是 Android 所使用的一種新的虛擬機。與 Dalvik 的主要不同在於：Dalvik 採用的是 Just-In-Time(JIT)技術，而 ART 採用 Ahead-of-time(AOT)技術。ART 同時也改善了性能、垃圾回收(Garbage Collection)、應用程式除錯以及性能分析。

2.1.1.2.2. Android 版本 App 是由核心 Android 作業系統擴展。

App 有 2 個主要來源：

●預載的行動應用 App

Android 包括一組預先安裝的 App，包括電話、電子郵件、日曆、網頁瀏覽器及聯繫人。這些行動應用 App 可讓使用者的可直接使用 App 或提供一種可以被其他 App 存取關鍵設備資源的能力。

預裝的 App 有些是開源 Android 平台的一部分，或者可以是由手機製造商(OEM)為特定行動智慧裝置所開發，以便運用其硬體獨特規格客製化功能(預載行動應用 App 安全非本指引範圍)。

●使用者安裝的 App

Android 提供了一個開放的開發環境，支援任何第三方 App。Google Play 提供超過數百萬種 App 可供使用者下載安裝。

2.1.1.2.3. Google 提供了一套可用於任何相容的 Android 設備基於雲端的服務之主要服務平台

●Google Play 商店

Google Play 是讓使用者能夠找尋、安裝，並從 Android 設備或網路購買 App 的集合的服務。Google Play 可讓開發人員可以輕鬆接觸到 Android 使用者和潛在客戶。Google Play 也提供社群審查，申請許可證驗證，App 安全掃描和其他安全服務。

●Android 更新

Android 的更新服務提供新的功能和安全更新到 Android 設備，包括通過網路或通過空中下載(OTA)更新。

●應用服務

允許 Android App 使用雲端功能框架，如(備份)App 資料和設置以及雲端到設備消息(C2DM)的推送消息。

2.1.1.2.4. 平台安全架構

- Android 尋求作為最安全和可用的行動作業平台，通過再重新強化傳統的作業系統的安全控制：
  - 保護使用者資料。
  - 保護系統資源(包括網路)。
  - 提供 App 隔離。
- 為了實現上述目標，Android 提供以下關鍵安全功能：(請參考 Android Security (查詢時間：2016/10/1)網站資訊)
  - 透過內核提供作業系統層級高安全性(詳見 System and kernel security)。
  - 強制所有 App 在個別沙箱內運行(詳見 System and kernel security)。
  - 安全的程序間通信(詳見 Application security)。
  - App 程式碼簽署(詳見 Trusty TEE)。
  - App 定義和使用者授予的權限(詳見 Application security)。
- 在基礎安全功能尚包括：
  - 身分認證(Authentication)

    支援硬體加密儲存密碼、認證閘道及指紋辨識認證。
  - 全磁碟加密(Full Disk Encryption)

    全磁碟加密是使用加密密鑰對 Android 設備上的所有使用者資料進行編碼的過程。一旦設備被加密，所有使用者創建的資料，

儲存到磁碟之前自動加密，並在返回到呼叫程序之前所有讀取自動解密資料。加密算法是 128 位元的高級加密標準(AES)與密碼塊鏈接(CBC)，簽章的加密雜湊函式為 ESSIV：SHA256。主密鑰是通過呼叫 OpenSSL 函式庫採用 128 位元 AES 加密。實作必須使用 128 位元以上的密鑰(256 位元亦可選用)。

– 安全強化 Linux(Security-Enhanced Linux in Android)

Android 的安全模型是基於在 App 沙箱的概念的一部分。每個 App 都在自己的沙箱中運行。在此之前的 Android 4.3，這些沙箱是由每個 App 在安裝時定義了獨特的 Linux 的 UID 所識別。

由搭載 Android 4.3 版本開始，安全增強型 Linux(SELinux)被用於進一步定義了 Android App 沙箱的邊界；作為 Android 安全模型的一部分，使 Android 運用 SELinux 對所有程序實施強制存取控制(MAC)，甚至與處理 root/su 權限(即 Linux 的能力)的運行。SELinux 的通過限制特權程序和自動化的安全策略創建增強 Android 安全性。

– 驗證啟動(Verified Boot)

Android 4.4 及更高版本支援可選用的 device-mApper-verity (dm-verity)的內核功能，提供了透明的完整性區塊設備的檢查驗證啟動。dm-verity 有助於能守住 root 權限和防止設備遭隱匿於深層 rootkit 的破解。

這一功能可以幫助 Android 使用者確認自從上次最後使用到現在，裝置是否持續處在相同的狀態？具有 root 權限的惡意軟體可以很智慧地將自己隱藏來，躲避檢測程序且偽裝為正常程序。

有 root 權限軟體可以輕易做到這一點，且往往比檢測工具有更多特權，使軟體能夠「欺騙」檢測程序。

2.1.1.3. 安全維護計畫

在發展 Android 平台早期，核心開發團隊已意識到維運一個強大的開源行動作業平台，需要創造一個眾多 App 和多廠牌行動智慧裝置硬體設備，及持續不間斷發展安全的 Android 平台與雲端服務支援的蓬勃生態系統。在 Android 團隊長久努力下，Android 平台在其整個開發生命週期，已經受到專業的安全計畫保護。

Android 團隊有機會觀察到眾多行動智慧裝置、桌上型電腦和伺服器平台如何防止或回應遭遇安全問題，並建立了一個以解決觀察到的 Android 平台弱點安全機制。

Android 的安全維護計畫的關鍵組成部分包括：

●設計審查

從 Android 平台開發生命週期的早期，Android 平台即設計了安全流程與創建豐富和可配置的安全模型。該平台的每個主要功能均由工程和安全資源審查，有適當的安全控制整合到系統的體系結構。

●滲透測試和程式審查

在平台的開發過程，Android 核心系統所創建和來自開源的元件均受到大量的安全審查。這些審查是由 Android 的安全團隊、Google 的資訊安全工程團隊及獨立安全顧問執行。這些審查的目的是找出 App 薄弱環節和潛在漏洞以及開源平台既有漏洞，在正式版本釋出前將由外部安全專家模擬分析這些類型的漏洞。

●開源與社群評論

Android 開源專案均經過任何有官方都能夠參與的廣泛的安全審查活動。Android 版還使用已經過嚴格外部安全審查的開源技術，如 Linux 內核。Google Play 為使用者和企業提供有關特定 App 提供一個論壇，可使用者的直接獲得資訊。

●安全事故應變流程

即使有以上的安全控制措施，上架後還是可能出現安全問題，這也是為什麼 Android 專案建立了一個全面的安全事故應變流程。Android 安全團隊全天候不斷監控的 Android 特有的和潛在漏洞的一般安全討論社群。一旦合法地發現問題，Android 團隊以既有應變流程，使安全漏洞的快速緩解，以確保潛在的風險對所有 Android 使用者衝擊最小化。這些作為也包括在雲端的支援與變更。

2.1.2. iOS 作業系統安全機制

2.1.2.1. 作業系統簡介

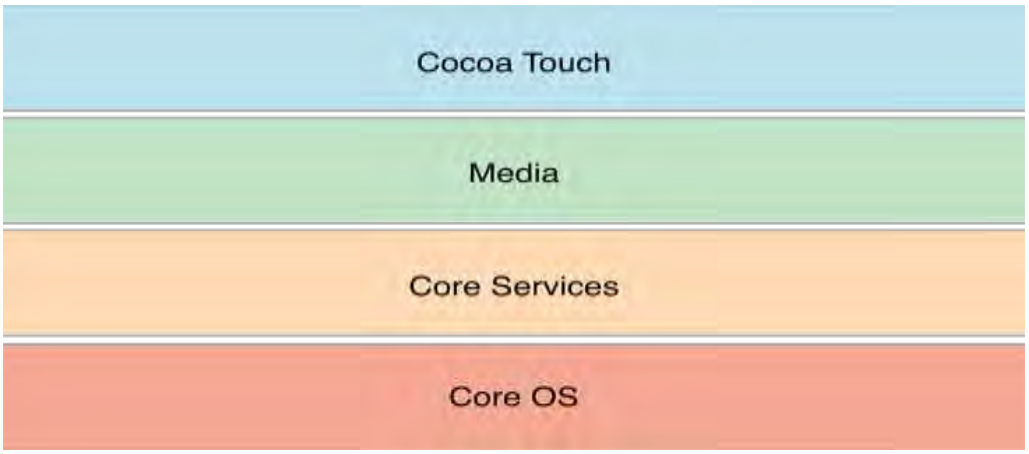
2.1.2.1.1. iOS 系統專屬性

iOS(原名 iPhone OS)是 Apple 公司為行動裝置所開發的封閉作業系統，所支援的裝置包括 iPhone、iPod touch、iPad 及 Apple TV。與 Google Android 及 Windows Phone 不同，iOS 不支援任何非 Apple 的硬體裝置。

2.1.2.1.2. iOS 系統架構

●框架

iOS 使用基於與 Apple 麥金塔(Macintosh)電腦不同的 ARM 架構的 64 位元或 32 位元之中央處理器，使用以 Darwin 作為基礎的系統架構層次：iOS 分為核心作業系統層(the Core OS layer)、核心服務層(the Core Services layer)、媒體層(the Media layer)及觸控應用層(the Cocoa Touch layer)。



資料來源：Apple Developer Portal（查詢時間：2016/10/1)

圖3 iOS 系統層級

●使用者介面

– 設計特點

iOS 使用者介面能使用按鍵、多點觸控對裝置進行控制。多點觸控：Apple 稱為 Multi-Touch，包括滑動(Swiping)、輕按(Tapping)、擠壓(Pinching)、反向擠壓(Reverse Pinching or unpinching)，於 iPad 中還可使用四指或五指手勢切換 App，與系統互動。

– 螢幕介面

以 App 方格的形式呈現，可將多個程式合併至一個資料夾方格中；最底部的一欄為 Dock，最多可以有 4 個(iPad 系列產品則為 6 個)。

●App 商店(App Store)

App Store：App Store 是 Apple 公司為旗下作業系統所建立和維護的數位化應用發布平台，允許用戶從 iTunes Store 瀏覽和下載一些由 iOS SDK 或者 Mac SDK 開發的 App。根據 App 發布的不同情況，使用者可以付費或者免費下載。App 可以直接下載到 iOS 裝置，也可以透過 mac OS 或者 Windows 將 iTunes 下載到電腦中。

2.1.2.1.3. iOS 開發人員計劃

●iOS 開發人員計劃(iOS Developer Program 查詢時間:2016/10/1)是 Apple 公司為 iOS 開發人員提供的官方計畫

該計畫包括為開發人員提供開發工具、技術支援培訓、資格及程式發布稽核等支援系統。Apple 公司的 iOS 開發人員中心網站亦提供了大量技術和學習資料。此外 Apple 公司每年都會舉辦 Apple 公司全球軟體開發人員年會。

開發人員帳號分為個人與公司組織，公司組織需要實名申請。個人帳號費用為每年 99 美元，公司帳號為每年 299 美元。需要有帳號才能將 App 上傳商店，若帳號過期或失效，App 也會被下架。App 的付費收入(含定價及 In-App-Purchase)，收取 30%費用，剩下的 70%給予開發人員。

●iTunes Connect

針對每個 App 採實質審核，開發人員必須將 App 與各項資料都輸入完成，除了軟體上傳時會審核是否有使用非法 API 以外，審查人員都會進行實質功能審核(參照審查指南 (查詢時間：2016/10/1))，初次審核所需時間約為 2 周，之後每次更新也須審查，約為 1 周。

●開發工具(iOS 軟體開發套件)

iOS 軟體開發套件(iOS SDK(Software Development Kit)亦稱：iPhone SDK)是由 Apple 公司開發的為 iOS 設計的 App 開發工具包。首個版本於 2008 年 2 月發布。軟體開發套件需要在 Mac OS X Leopard 及以上系統並擁有英特爾處理器才能執行，其他的作業系統，包括微軟的 Windows 作業系統和舊版本的 Mac OS X 作業系統都不被支援。

自從 Xcode 3.1 發布以後，Xcode 就成為了 iOS 軟體開發套件的開發環境。和 Mac OS X 的 App 一樣，iOS App 使用 Objective-C 語言，一些 App 可以寫成 C 或 C++語言。另外，在 2014 年 6 月發表 Swift 語言，可採用於 Xcode 之上，是下一代開發語言，提供了不少的技術改善。

●裝置越獄(iOS Jailbreaking)

iOS 裝置越獄是用於取得 Apple 公司行動裝置作業系統 iOS 最高權限(類似 Android 的 root)的一種技術手段。越獄完成後，裝置將執行帶有最高權限的 iOS 作業系統，一般來說越獄工具會輔助在已獲取權限的 iOS 環境下安裝一款名為 Cydia 的軟體。

Cydia 是一款由開發人員 Jay Freeman 主導開發的 iOS 第三方外掛模組應用商店，通過此軟體可以完成越獄前不可能進行的動作，

例如：安裝 App Store 以外的軟體、更換外觀主題、執行 Shell 程式及甚至可能解開運營商對手機網路的限制(即俗稱的「解鎖」)。

Jay Freeman 在 2010 年 10 月估計，全球大概有 10%的 iPhone 曾進行過越獄。越獄行為除了侵犯 App 商店的銷售權利、破解合法軟體以外，也造成該裝置的安全問題。

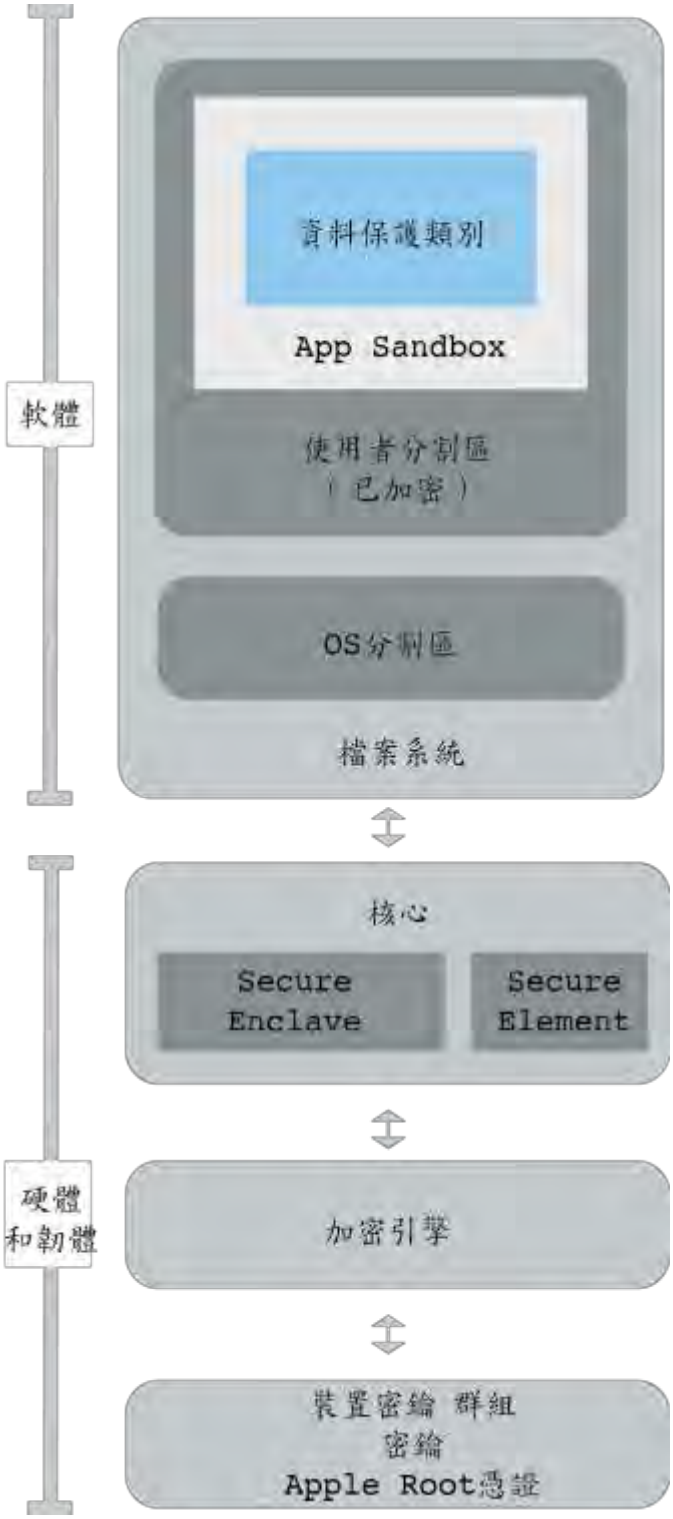
2.1.2.2. 安全功能簡介

2.1.2.2.1. 硬體、韌體、作業系統

軟體、硬體和服務在每台 iOS 裝置上緊密合作，其安全架構詳見圖 4 所示，一同為使用者提供最高的安全性和直接的使用者體驗。iOS 不僅保護裝置和其中的靜態資料，同時也保護了整個生態系統，包含使用者在本機、網路上以及使用重要 Internet 服務所執行的所有操作。

iOS 和 iOS 裝置不僅提供進階的安全性功能，而且還容易使用。許多安全性功能預設便已啟用，因此 IT 部門無須執行大量的設定動作。而如裝置加密之類的重要安全性功能則無法設定，因此可避免使用者不小心地停用這些功能。

啟動程序中每個步驟包含的元件都經過 Apple 加密簽署以確保其完整性，且只有 在驗證信任鏈結後，每個步驟才能繼續。這包含 bootloader、核心、核心延伸功能及基頻韌體。



資料來源：[iOS 安全指引（查詢時間：2016/10/1)]

圖4　iOS 的安全架構圖

●安全啟動鍊(Secure boot chain)

開啟 iOS 裝置後，其 App 處理器會立即執行唯讀記憶體(稱為 Boot ROM)中的程式碼。此類無法更改的程式碼(稱為硬體的信任 root)是在製造晶片時完成設定，且已間接獲得信任。

Boot ROM 程式碼包含 Apple Root CA 公用密鑰，該公用密鑰用來驗證 Low-Level Bootloader(LLB)是否經過 Apple 簽署，以決定是否允許其載入。

這是信任鏈結中的第一步，信任鏈結中的每個步驟都會確保下一個步驟經由 Apple 簽署。當 LLB 完成其任務後，便會驗證和執行下一階段的 bootloader(即 iBoot)，接著會驗證並執行 iOS 核心。

此安全啟動鏈有助於確保底層的軟體未經竄改，並只允許 iOS 在經過驗證的 Apple 裝置上執行。

對於具有行動網路連線功能的裝置，基頻子系統也會利用其類似的安全啟動程序，包含已簽署的軟體以及由基頻處理器驗證的密鑰。

對於搭載 A7 或更新版本 A 系列處理器的裝置，Secure Enclave 副處理器也會利用安全啟動程序，以確保其獨立的軟體經過 Apple 驗證和簽署。

若此啟動程序的某個步驟無法載入或驗證下一個程序，啟動就會停止，裝置螢幕上會顯示「連接 iTunes」。這即是所謂的恢復模式。若 Boot ROM 無法載入或驗證 LLB，系統會進入 DFU(裝置韌體升級)模式。在這兩種情況下，裝置都必須透過 USB 連接 iTunes，並回復到出廠設定。

●系統軟體授權(System Software Authorization)

Apple 定期發布軟體更新來解決新出現的安全問題並且還提供了新的功能；提供所有支援設備的這些更新同時。用戶接收設備上，並通過 iOS 的更新通知 iTunes 和更新無線傳送，鼓勵迅速採用最新安全修補程序。

上述啟動過程有助於確保只有 Apple 簽名的程式碼可以安裝在設備上。為了阻止設備被降級到舊版本缺乏最新的安全更新，iOS 設備使用一種稱為系統軟體授權過程。

如果系統允許降版本，攻擊者取得設備後，就可以安裝舊版本，以便透過這些已經被新版本修正的漏洞，攻擊此設備。

在與 A7 或更高版本的 A 系列處理器，配合安全領地副處理器，實現系統軟體的授權，以確保軟體的完整性和防止降級安裝。

iOS 軟體更新可使用 iTunes 安裝，或在裝置上採用無線方式(OTA)進行安裝。若使用 iTunes，系統會下載並安裝完整的 iOS 拷貝。若採用 OTA 方式安裝軟體更新，系統將只會下載完成更新所需的元件(而非下載整套 OS)，以改善網路效率。此外，可以在執行 OS X Server 快取服務的區域網路伺服器上快取軟體更新，這樣 iOS 裝置無須連接 Apple 伺服器便可取得必要的更新資料。

在 iOS 升級期間，iTunes(若採用 OTA 軟體更新方式，則為裝置本身)會連接 Apple 安裝授權伺服器，並向其傳送以下資料：要安裝之安裝套件中的各部分加密 測量值列表(如 LLB、iBoot、核心及 OS 映像檔)、隨機反重播的值(隨機數)以及裝置的唯一識別碼(ECID)。

授權伺服器會將提供的測量值列表與允許安裝的版本進行比較，若

找到相符項目，便會將 ECID 加入到測量值中並對結果進行簽署。伺服器會將完整的一組已簽署資料傳遞至裝置，這是升級程序的一部分。加入獨一無二 ECID 可作為裝置「個人化」授權作業要求。藉由只對已知的測量值授權和簽署，伺服器可確保更新的內容與 Apple 所提供的完全相同。

啟動時的信任鏈結評估會驗證簽名是否來自 Apple，並確認從磁碟載入的項目測量值在結合裝置 ECID 後，是否與該簽名所涵蓋的內容相符。

這些步驟可確保授權是針對特定裝置進行，並且舊版 iOS 無法從一部裝置拷貝到另一部裝置。隨機數可阻止攻擊者儲存伺服器的回應，並阻止使用該回應來破壞裝置或以其他方式竄改系統軟體。

- Secure Enclave 副處理器

Secure Enclave 是 Apple A7 或更新版本 A 系列處理器精心打造的副處理器，使用加密記憶體，並包含一個硬體亂數產生器。Secure Enclave 為「資料保護」密鑰管理提供所有加密操作，即使在核心遭到入侵的情況下，也可維護「資料保護」的完整性。Secure Enclave 與 App 處理器之間的通訊會被隔離到一個已中斷驅動的信箱和共享的記憶體資料緩衝區。

Secure Enclave 會執行 Apple 自定版本的 L4 微核心系列。Secure Enclave 可利用其本身的安全啟動，使用不同於 App 處理器的個人化軟體更新程序進行更新。

每個 Secure Enclave 在製作期間皆已提供其本身的 UID(唯一識別碼)，此 UID 無法由系統的其他部分取用，且 Apple 亦無其相關

資訊。當裝置啟動時，會製作一個臨時密鑰，此密鑰與 UID 配合使用，用來對 Secure Enclave 的裝置記憶體空間部分進行加密。

另外，由 Secure Enclave 儲存到檔案系統的資料會藉由 UID 搭配使用的密鑰和反重播計數器來進行加密。

Secure Enclave 負責處理來自 Touch ID 感應器的指紋資料，確定是否有與登記之指紋相符合的指紋資料，然後代表使用者啟用存取或購買。處理器與 Touch ID 感應器之間的通訊是透過序列週邊介面匯流排來執行。處理器會將資料轉送至 Secure Enclave，但處理器本身無法讀取這些資料。資料會藉由區段密鑰進行加密與認證，該密鑰透過為 Touch ID 感應器和 Secure Enclavek 布建的裝置共享密鑰來進行交涉。區段密鑰的交換會針對雙方使用 AES 密鑰封裝，並提供一個用來建立工作階段密鑰和使用 AES-CCM 傳輸加密的隨機密鑰。

2.1.2.2.2. 周邊設備(HomeKit/HealthKit/Watch)

周邊設備如 HomeKit/HealthKit/Watch 這 3 類，在通訊、資料傳輸、共用、備份及輸出等機制，也都配合硬體與軟體的簽章、加密及保護，開發人員只需依照所提供的規格進行開發整合即可。請參考 [healthkit - Apple developer (查詢時間：2016/10/1)]。

2.1.2.2.3. 加密與資料保護

- 密碼(Password/TouchID/ID)

藉由設定裝置密碼，使用者可以自動啟用「資料保護」。iOS 支援 6 位數、4 位數和任意長度的英數字元密碼。除了用於解鎖裝置外，密碼還為特定加密密鑰提供熵。這表示攻擊者即使拿到裝置，在沒有密碼的情況下也無法取用特定保護類別中的資料。

密碼與裝置的 UID 搭配使用，因此暴力密碼破解只能在受攻擊的裝置上進行。因此，iOS 系統使用較大的反覆運算來延緩每次的嘗試。反覆運算計數已經過測定，每次嘗試會耗時約 80 毫秒。這意味著嘗試 6 個字元的英數字元密碼(小寫字母和數字)的所有組合將會耗時 5 年半的時間。

密碼錯誤次數	1-4	5	6	7-8	9
需等待分鐘數	0	1	5	15	60

●資料保護類別

在 iOS 裝置上製作新檔案時，用來製作的 App 會替檔案指定一個類別。每個類別使用不同的規則來決定資料何時可供取用。

請參考啟用資料保護- Apple developer。

●鑰匙圈資料保護

許多 App 需要處理密碼和其他簡短但較為敏感的資料，如密鑰和登入 Token。iOS 鑰匙圈提供了儲存這些項目的安全方式。

鑰匙圈項目只能在來自同一開發人員的 App 間共享。管理方式是要求第三方 App 使用取用群組，並使用透過 Apple Developer Program(Apple 開發人員計畫)或透過應用程式群組來為其分配前置碼。對前置碼的要求和應用程式群組唯一性，是透過程式碼簽署、佈建描述檔和 Apple Developer Program(Apple 開發人員計畫)強制執行。

請參考 Keychain concepts - Apple developer。

●App 安全性(App Security)

App 是現代行動安全架構最關鍵的要素之一。雖然 App 可顯著提高使用者的生產力，但若處理不當，也可能對系統安全性、穩定性和使用者資料產生負面影響。

有鑑於此，iOS 提供了多重保護來確保 App 經過簽署和驗證，且以沙箱(Sandbox)技術限制，進而保護使用者資料。這些要素為 App 提供了穩定且安全的平台，讓成千上萬的開發人員能夠在 iOS 上提供數十萬款的 App，而不會影響系統的完整性。使用者可以在其 iOS 裝置上取用這些 App，無須過度擔心病毒、惡意軟體或未經授權的攻擊。

●App 程式碼簽署(App Code signing)

若要在 iOS 裝置上開發並安裝 App，開發人員必須向 Apple 註冊並加入 Apple Developer Program(Apple 開發人員計畫)。Apple 會先驗證每位開發人員(無論是個人或企業)的真實身分，然後再核發憑證。

開發人員可使用該憑證對 App 進行簽署，並將其上傳至 App Store 進行發布。因此，App Store 中的所有 App 都是由身分可識別的個人或組織上傳的，藉此阻止製作惡意 App。

●執行階段程序安全性(Runtime process security)

一旦確認 App 來自核准的來源後，iOS 會強制執行相關的安全措施，以防止其危害其他 App 或系統的其他部分。

所有第三方的 App 均會以 Sandbox 技術限制，因此在存取其他 App 儲存的檔案或對裝置進行更動時會受到限制。這樣可以防止 App 蒐集或修改其他 App 儲存的資訊。每個 App 都有唯一的主目錄來存放其檔案，主目錄是在安裝 App 時隨機指定的。如果第三方的

App 需要存取除了本身資訊以外的其他資訊，只能透過 iOS 明確提供的服務來執行。

系統檔案和資源也會與使用者的 App 保持區隔。iOS 的大部分操作與所有第三方的 App 一樣，以非特殊權限使用者「mobile」的身分執行。整個作業系統分割區都裝載為唯讀。不必要的工具(如遠端登入服務)並未包含在系統軟體中，並且 API 不允許 App 提升自己的特殊權限來修改其他 App 或 iOS 本身。

●延伸功能(Extensions)

iOS 透過延伸功能來對其他 App 增加功能。延伸功能是具有特殊用途的已簽署可執行二進位程式碼，封裝在 App 內。系統會在安裝時自動偵測延伸功能，並讓使用相符系統的其他 App 使用這些延伸功能。

●App 群組(App Groups)

在 App 被設定為「App 群組」的一部分後，便可存取以下內容：

– 磁碟上共享的儲存容器，只要 App 群組內有一個 App 被安裝，就會一直保留在裝置上。

– 共享的偏好設定。

– 共享的鑰匙圈項目。

2.1.2.2.4. 網路安全性 ATS(TLS)/VPN/Single Sign-On

●TLS

iOS 支援傳輸層安全性(TLS v1.0、TLS v1.1、TLS v1.2)和 DTLS。

●VPN

對於使用以憑證為基礎的認證網路，iOS 支援「隨選即用 VPN」。IT 規則會藉由使用設定描述檔來指定哪些網域需要 VPN 連線。

●單一登入

iOS 支援透過單一登入(SSO)對企業網路進行認證。SSO 搭配以 Kerberos 為基礎的網路使用，針對使用者獲授權取用的服務對使用者進行認證。SSO 可用於各種網路活動，從安全的 Safari 區段到第三方的 App。

●Apple Pay

使用 Apple Pay，使用者可以使用受支援的 iOS 裝置和 Apple Watch 來以簡單、安全又保密的方式來進行付款。對於使用者來說很容易，且在硬體和軟體方面皆具備整合性的安全措施。

Apple Pay 的目標也在於保護使用者的個人資訊。Apple Pay 不會蒐集任何可追蹤使用者的交易資訊。付款交易僅於使用者、商家和發卡機構之間流通。

2.2.行動應用 App 開發環境(Android/iOS)

2.2.1. Android 開發環境簡介

Android Studio 是 Google 為 Android 平台開發程式的整合式開發環境。2013/5/16 在 Google I/O 上發布，可供開發人員免費使用。在此之前 Android 開發人員大部分都是以 Java 的 Eclipse 為整合式開發環境，再加裝 Android SDK，現今 Android 也支援 Eclipse ADT 開發外掛程式，本指引以介紹 Android Studio 為主，如需進一步了解 Eclipse 整合式開發環境，請參考 Eclipse,Wiki。將 Android Studio 和 Eclipse ADT 比較詳見表 2 所示。

表2 Android Studio 和 Eclipse ADT 比較

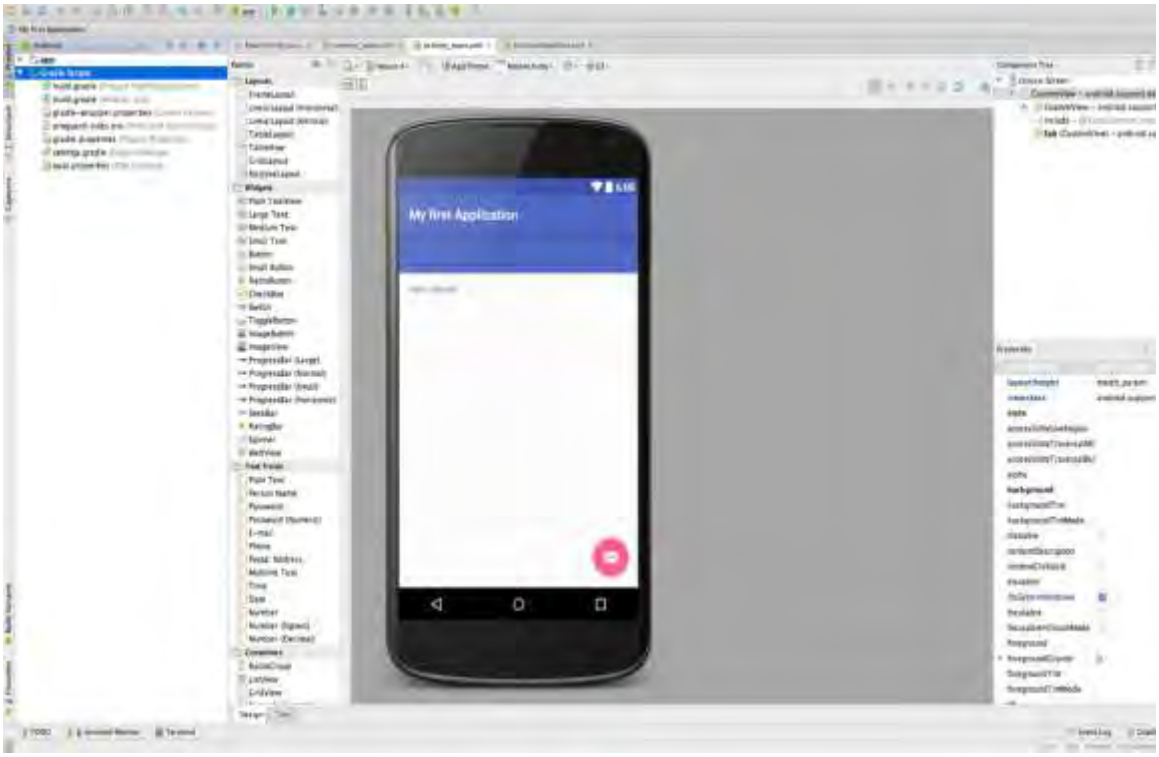
特性	Android Studio	Eclipse ADT
編譯系統	Gradle	Apache Ant
基於 Maven 的建構依賴	是	否
建構變體(Build Variants)和多 APK 產出	是	否
高階的 Android 程式碼完成和重構	是	否
圖形布局編輯器	是	是
APK 簽名和金鑰庫管理	是	是
NDK 支援	是	是

資料來源：https://zh.wikipedia.org/wiki/Android_Studio (查詢時間：2016/10/1)

Android Studio 於 2013 年 5 月發布早期預覽版本，版本號為 0.1。2014 年 6 月發布 0.8 版本，至此進入 beta 階段。第一個穩定版本 1.0 於 2014/12/8 發布。2016 年 8 月最新版本為 2.1.2。

Android Studio 基於 JetBrains IntelliJ IDEA，為 Android 開發特殊客製，並在 Windows、OS X 和 Linux 平台上均可執行(開發介面詳見圖 5 所示)。

下載及安裝請參考：Android Studio 官方網頁。(查詢時間：2016/10/1)



資料來源：本計畫整理

圖5 Android Studio 整合式開發環境

2.2.1.1. 主要功能

- 視覺化布局：WYSIWYG 編輯器-即時編碼-即時程式介面預覽。
- 開發人員控制台：最佳化提示，協助翻譯，來源跟蹤，宣傳和行銷曲線圖-使用率度量。
- Beta 版本測試，並階段性展示。
- 基於 Gradle 的建構支援。
- Android 特定程式碼重構和快速修復。
- Lint 提示工具可以更有效地對程式效能、可用性、版本相容及其他問題進行控制捕捉。
- 支援 ProGuard 和應用簽名功能。

- 基於模板的精靈來產出常用的 Android 應用設計和元件。
- 內建布局編輯器，可讓開發人員拖放 UI 元件，並預覽在不同尺寸裝置上的 UI 顯示效果等。
- 支援建構 Android Wear 應用。
- 內建 Google Cloud Platform，支援 Google Cloud Messaging 和 App Engine 的整合。

2.2.1.2. 系統安裝要求

表3 安裝 Android Studio 系統要求表

	Windows	OS X	Linux
作業系統版本	Microsoft Windows 10/8.1/8/7/Vista/2003 (32 或 64 位元)	OS X 10.8.5 或更高版本，最高 10.10.5 (Yosemite)	GNOME、KDE、Unity desktop on Ubuntu、Fedora、GNU/Linux Debian
記憶體	最低 2 GB，推薦 4 GB 記憶體		
磁碟空間	500 MB 磁碟空間，至少 1 GB 用於 Android SDK，模擬器系統映像和快取		
JDK 版本	Java Development Kit (JDK) 7 或更高版本		
螢幕解析度	最低 1280×800 螢幕解析度		

資料來源：https://zh.wikipedia.org/wiki/Android_Studio (查詢時間：2016/10/1)

2.2.1.3. 開發人員入口網

Google 為開發人員提供了相當完備的開發人員入口網(查詢時間：2016/10/1)，包含了設計、開發及發布 3 大部分，在開發環境(查詢時間：2016/10/1)中提供：

●訓練課程

包含多項載具平台、使用介面、資料儲存、與其他 App 互動等的原理及示範程式碼。

●API 指南

提供內容豐富的 App 架構，並了解如何使用 Android 的各種 API 建置 App。

●參考資源

提供各 API 的 Class、Method 及參數的檢索。

●程式碼範例

提供各類功能範例檔。

●Android Studio

整合式開發環境。

●Android NDK

NDK(Native Development Kit)是 Android 開發用的工具集，和 SDK 不同的是，NDK 允許開發人員使用 Android 的 C/C++原生語言，且可與 SDK 的 Java 語言一同使用，實作出 App。

●Google Services

包含了 In-App billing、In-App Subscriptions 及 Google Play Developer API 等與付費、購買、授權 App 的 API 及發布 APK 等支援。

2.2.1.4. Google Play 開發人員控制台

Google Play 開發人員控制台(Google Play Developer Console)：是提供行動應用 App 上架、行銷及後續追蹤的控制台，需要付費註冊開發人員帳號才能使用，主要提供：

●上傳 App。

●設定價格。

●使用 Alpha 和 Beta 測試功能。

●查看報表、統計資料和評論。

●查看 App 效能表現統計資料。

●查看 App 的評分和評論。

●查看 App 收益資料。

Android Studio 還有一項重要的參考資源是「用戶指南(User Guide)」(查詢日期：2016/10/1)，除了介紹基本操作環境，並提供從 Eclipse 轉移的步驟並建議了基本開發流程，區分以下 5 個基本流程:

●設置工作區。

●撰寫 App。

●建構及執行。

●除錯、配置和測試。

●發布 App。

2.2.1.5. API 等級(API Level)

API 等級是一個整數值，用來識別一個 Android 平台版本所提供的 API 框架版本。

Android 平台提供了一個 API 的框架，App 可以使用它與底層的 Android 系統進行互動。該 API 框架包括：

●一套核心包(Packages)和類(Class)。

●一種聲明 manifest 文件組 XML 元素和屬性。

●一個聲明和存取資源組 XML 元素和屬性。

●一組的意圖。

●一組 App 能夠請求權限，以及包括在系統中允許執行權限。

Android 平台的後續版本可以包括更新 Android App 框架的 API。

更新到框架的 API 被設計，以使新的 API 保持與較早版本的 API 相容。

該 API 框架，一個 Android 平台提供使用稱為「API 等級」的整數值指定。每一個 Android 平台的版本完全相同支援一個 API 等級，雖然支援是隱含的所有早期的 API 等級(最低 API 等級 1)。
Android 平台的初始版本提供的 API Level 1 和後續版本都增加了 API 等級。

Android 平台正在運行的每個版本的資訊，請參閱平台版本的儀表板(查詢日期：2016/10/1)。

App 開發人員、品管測試及資安人員等應充分了解 Android 行動應用平台提供安全功能(參考 2.1.1 Android 作業系統安全機制)及 Android 提供開發的資源，以利在第 3 章的安全開發實務找到需要的資源及實作的參考。

表4 Android 各版本支援 API 等級

平台版本	API Level	VERSION_CODE	註解
Android 7.0	24	Nougat	平台特點
Android 6.0	23	Marshmallow	平台特點
Android 5.1	22	LOLLIPOP_MR1	平台特點
Android 5.0	21	LOLLIPOP	
Android 的 4.4W	20	KITKAT_WATCH	只有 KitCat 的穿戴式
Android 4.4	19	KITKAT	平台特點
Android 4.3	18	JELLY_BEAN_MR2	平台特點
Android 4.2、4.2.2	17	JELLY_BEAN_MR1	平台特點
Android 4.1 、4.1.1	16	JELLY_BEAN	平台特點
Android4.0.3、4.0.4	15	ICE_CREAM_SANDWICH_MR1	平台特點
Android 4.0：4.0.1、4.0.2	14	ICE_CREAM_SANDWICH	
Android 3.2	13	HONEYCOMB_MR2	
Android 3.1.X	12	HONEYCOMB_MR1	平台特點
Android 3.0.x	11	HONEYCOMB	平台特點
Android 2.3.4、Android 2.3.31	10	GINGERBREAD_MR1	平台特點
Android 2.3.2、Android 2.3.1、Android 2.3	9	GINGERBREAD	
Android 2.2.x	8	FROYO	平台特點
Android 2.1.x	7	ECLAIR_MR1	平台特點
Android 2.0.1	6	ECLAIR_0_1	
Android 2.0 的	5	ECLAIR	
Android 1.6	4	DONUT	平台特點
Android 1.5	3	CUPCAKE	平台特點

平台版本	API Level	VERSION_CODE	註解
Android 1.1	2	BASE_1_1	
Android 1.0	1	BASE	

資料來源：https://developer.android.com/guide/topics/manifest/uses-sdk-element.html (查詢時間：2016/10/1)

2.2.2. iOS 開發環境簡介

2.2.2.1. 官方開發 IDE(Xcode)說明

2.2.2.1.1. Xcode 開發、上傳流程

XCode 是由公司開發設計的整合開發環境(IDE)，軟體開發工具程式，目前的版本是 7.x 版。XCode 目前都是免費軟體，可在 MacAppStore 市集進行下載。

The Xcode suite 包含有 GNU Compiler Collection 自由軟體，並支援 C 語言、C++、Fortran、Objective-C、Objective-C++、Java、AppleScript、Python、Ruby 和 Swift，還提供 Cocoa、Carbon 以及 Java 等編程模式。Xcode 使用 GDB 作為其後台偵錯工具。XCode 開發說明如下：

●MVC 架構

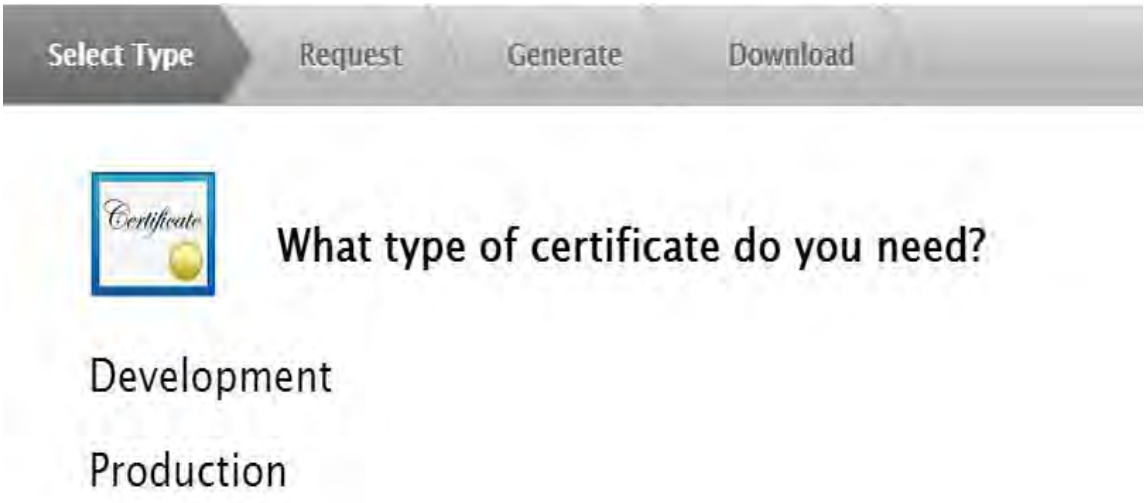
Xcode 原則上以 MVC 架構進行開發：View 視圖為 UIView 類別為基底。Controller 控制器為以各種 Controller 類別為基底。而 Model 模型：用於封裝與應用程序的業務邏輯相關的資料以及對資料的處理方法，直接和資料做為連結，記錄了要執行的動作與實作的方法。由 iOS Framework 為主要提供者。

●Apple 開發人員入口網

Apple 提供各種資源給開發人員，並對於 App 進行控管，若使用者加入開發人員之列，由 Apple 開發人員入口網 (https://developer.Apple.com)登入後，就可以管理進行開發所需要的憑證(Certificates)、裝置(Devices)、識別名稱(Identifiers)及裝置描述檔(Provisioning Profile)。

– 憑證

憑證簡單的說就是證明 App 開發人員的身分。當 App 開發人員在 iOS 開發人員計劃中註冊帳號後即可以取得，分為開發(Development)及產品(Production)兩大類的憑證。前者可讓 App 開發人員將 App 部署到裝置上(上限為 100 台)做測試；後者則是用來發佈到 App Store。時效為一年，若憑證被取消，則衍生的項目也會失效。



資料來源：developer.Apple.com

圖6　憑證建立流程與種類

– 裝置

Apple 提供 App adhoc 測試時，可安裝到實際裝置，但每個帳號只提供 100 個測試裝置可以登錄，每年更換開發人員授權時，可進行重置/修改裝置清單。

– App ID(identifier)

該識別名稱又稱為 App ID，即為 App 唯一的識別名稱，App ID 應與 Xcode 中的 Bundle ID 一致。App ID 分為 2 種，一種稱為 ExplicitAppID，即唯一的 App ID，用來對單一 App 命名；另一種稱為 WildcardAppID，類似使用正規式匹配符號的概念，指某些符合命名的 App，例如：可用*(星號)表示所有 App。

– Provisioning Profile

係指在 Xcode 或其他開發環境進行測試或正式發布時，所需使用的簽章憑證，用來確認 App 的合法性，其內容包含憑證(Certificates)、裝置(Devices)、識別名稱(Identifiers)。Profile 分成兩大類，分別為測試及產品用途。產品用途又分為 adhoc 與正式產品，adhoc 是指只能安裝在此帳號的裝置清單內的裝置上。

2.2.2.1.2. 偵錯模式/發布程式

在 Xcode 開發過程中有 2 種模式可供開發人員使用，分別是偵錯(Debug)與發布(Release)模式：

偵錯模式又可分為使用模擬器(iOS Simulator)與使用實際裝置執行。而發布模式：可分為 adhoc 模式與上架模式。以上模式都需要互相配合的 Provisioning Profile、AppID 及開發人員憑證等。

Adhoc 模式：產出的 IPA 檔案與 Provisioning Profile 交由測試者，使用 iTunes 工具程式進行安裝。上架模式的 IPA 檔案，則需使用 Application Loader 進行上傳準備上架。

2.2.2.1.3. 上架說明 iTunes 與付費說明(App/In-App)

登入 iTunesConnect 網站後，有 7 個分類功能，登入畫面詳見圖 7 所示。

資料來源：itunesconnect.com

圖7 iTunes Connect 登入後畫面

●協議、稅務與銀行業務

用於管理與 Apple 之間的合約、提供與 App 開發人員付款和扣繳稅款相關的必要財務資訊及追蹤 iTunes 協議的狀態。

●付款與財務報告

提供帳號之每月銷售金額、稅額及餘額報表。

●銷售與趨勢

提供帳號銷售之報表分析系統。可依區域、裝置、類別、內容類型、交易類型、完成狀態及版本等。

●我的 App

列出所有建立過的 App 清單，以管理 App 內容(售價/版本/開放地區/描述/說明)、更新、上下架及 IAP 付費(InAppPurchase)等維護。

●App 分析

提供 App/App 套裝之曝光次數、購買次數、銷售額、執行次數及當機次數等。

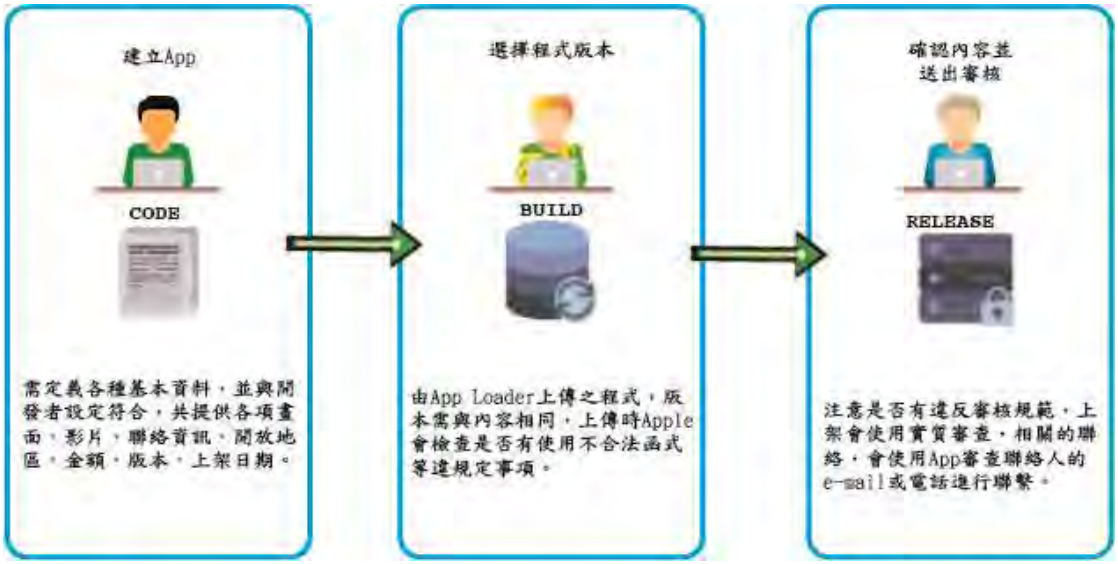
●使用者與職能

提供各項職能(法務/管理/財務/App 管理/開發人員/行銷人員/銷售報告)，可賦予職能給使用者，讓這些使用者分別檢視/管理各項工作。而人員又分為：iTunesConnect 使用者、TestFlight Beta 版測試人員及沙箱測試人員。

●資源與輔助說明

提供各項資源並輔助說明，以利相關人員使用與開發。

●上架說明詳見圖 8 所示。



資料來源：本計畫整理

圖8 itunes connect 上架步驟

2.2.2.2. 第三方開發商

2.2.2.2.1. Flex Builder(Adobe)

Flex Builder 是 Adobe 公司推出的軟體開發 IDE 產品，原本只能開發 Flash 程式，自從 2011 年推出的 4.5 版，就有支援 App 開發，可直接於該軟體開發 iOS 與 Android 平台的手機 App。

表5 Flex Builder 第三方開發平台優缺點

項目	使用 Flex Builder 開發 App
優點	可直接使用 Flex 元件開發，將 App 進行上傳
缺點	▪使用者介面非原生、使用者適應不佳。 ▪App 程式底層過大。 ▪與原生的介面、應用差異度過大，非直接使用原生 Framework。 ▪非原生開發：可能有潛在安全風險與無法對安全性修補。

資料來源：本計畫整理

2.2.2.2.2. Xamarin Stduio (Microsoft)

Xamarin Studio 是一個現代的，先進的 IDE 具有許多功能，用於創建 Xamarin.iOS，Xamarin.Mac，Xamarin.Android 和 Xamarin.Forms 應用。

包含了豐富的編輯、測試、與 iOS、Mac 和 Android 原生平台的整合和整合的源程式碼控制的名字。綜整優缺點詳見表 6 所示。

表6 Xamarin 第三方開發平台優缺點

項目	使用 Xamarin Stduio/Microsoft Visual Studio 開發 App
優點	▪可直接使用 C#語言進行開發跨平台 App(含 iOS/ Android/Microsoft Phone)。 ▪Xamarin Forms：可寫一套原始碼，應用於跨平台 App。 ▪支援原生開發人員功能：可寫作原生 App。
缺點	▪使用者介面非原生、使用者適應不佳。 ▪App 程式底層過大。 ▪雖號稱支援原生者開發功能，但程度與版本不一。 ▪仍非完全原生，可能有潛在安全風險、無法對安全性修補。

資料來源：本計畫整理

2.2.2.2.3. PhoneGap (Adobe Cordova)

PhoneGap 是一款開放原始碼的行動裝置開發框架，旨在讓開發人員使用 HTML、Javascript、CSS 等 Web APIs 開發跨平台的行動裝置 App。原本由 Nitobi 公司開發，現在由 Adobe Systems 擁有。

PhoneGap 是一個行動裝置的 API 介面集，利用 JavaScript 存取這些介面可以呼叫諸如攝影機、羅盤等硬體系統資源。配合上一些基

於 HTML5、CSS3 技術的 UI 框架，如 jQuery Mobile、Dojo Mobile 或 Sencha Touch，開發人員得以快速地開發跨平台 App 而不需要編寫任何的原生程式碼。

由於使用 Web 技術，PhoneGap 程式的載入和 UI 介面的反應都比原生的程式慢。Adobe 警告開發人員，由於使用 PhoneGap 框架開發的程式執行速度可能會太慢或使用體驗不夠「原生」，而被 Apple 應用商店拒絕上架。

綜整優缺點詳見表 7 所示；原生 API 程式庫列表詳見表 8 所示；著名第三方 API 程式庫詳見表 9 所示。

表7 PhoneGap 第三方開發平台優缺點

項目	使用開發 PhoneGap App
優點	▪可直接使用 Html 5 為基礎的語言進行開發跨平台 App。 ▪Html 5 Base：可寫一套原始碼，應用於跨平台 App。
缺點	▪使用者介面非原生、使用者適應不佳(類似使用網頁瀏覽器)。 ▪使用網頁技術，反應取決於網路等效能，使用者經驗較差。 ▪安全問題高，開發人員底層技術掌握度低。 ▪仍原生，可能有潛在安全風險、無法對安全性修補。

資料來源：本計畫整理

表8 原生 API 程式庫列表

名稱	可用版本	前置碼	描述
Accelerate.framework	4.0	cblas,vDSP	加速與 DSP 功能
Accounts.framework	5.0	AC	使用者帳號
AddressBook.framework	2.0	AB	通訊錄資料
AddressBookUI.framework	2.0	AB	通訊錄 UI

名稱	可用版本	前置碼	描述
AdSupport.framework	6.0	AS	廣告資料與蒐集
AssetsLibrary.framework	4.0	AL	圖片與影片
AudioToolbox.framework	2.0	AU,Audio	聲音串流與建立/撥放
AudioUnit.framework	2.0	AU,Audio	基本聲音播放
AVFoundation.framework	2.2	AV	錄音與撥放聲音/影片
AVKit.framework	8.0	AV	錄音與撥放聲音/影片
CFNetwork.framework	2.0	CF	網路與廣播取用
CloudKit.framework	8.0	CK	iCloud 資料取用
CoreAudio.framework	2.0	Audio	CoreAudio 相關類別
CoreAudioKit.framework	8.0	CA	CoreAudio 相關類別
CoreBluetooth.framework	5.0	CB	藍芽取用
CoreData.framework	3.0	NS	CoreData 類別
CoreFoundation.framework	2.0	CF	相關基本類別
CoreGraphics.framework	2.0	CG	2D 影像類別
CoreImage.framework	5.0	CI	圖片與影片取用
CoreLocation.framework	2.0	CL	GPS 地理資訊取用
CoreMedia.framework	4.0	CM	低階圖片與影片編輯
CoreMIDI.framework	4.2	MIDI	MIDI 格式音源處理
CoreMotion.framework	4.0	CM	加速器與陀螺儀取用
CoreTelephony.framework	4.0	CT	通話類別
CoreText.framework	3.2	CT	基本文字類別
CoreVideo.framework	4.0	CV	低階影像處理類別
EventKit.framework	4.0	EK	使用者日曆取用
EventKitUI.framework	4.0	EK	使用者日曆類別
ExternalAccessory.framewo	3.0	EA	外部硬體附件存取

名稱	可用版本	前置碼	描述
rk			
Foundation.framework	2.0	NS	低階資料處理
GameController.framework	7.0	GC	遊戲硬體控制類別
GameKit.framework	3.0	GK	點對點通訊類別
GLKit.framework	5.0	GLK	OpenGL 類別
GSS.framework	5.0	GSS	安全相關類別
HealthKit.framework	8.0	HK	健康裝置存取類別
HomeKit.framework	8.0	HM	家庭裝置存取類別
iAd.framework	4.0	AD	AD 廣告類別
ImageIO.framework	4.0	CG	影像處理存取類別
IOKit.framework	2.0	N/A	(內部使用)IO 存取類別
JavaScriptCore.framework	7.0	JS	Java-Script 基本類別
LocalAuthentication.framework	8.0	LA	指紋辨識類別
MapKit.framework	3.0	MK	地圖 UI 與座標類別
MediaAccessibility.framework	7.0	MA	媒體存取類別
MediaPlayer.framework	2.0	MP	媒體播放類別
MediaToolbox.framework	6.0	MT	媒體播放工具 UI 類別
MessageUI.framework	3.0	MF	電子郵件 UI 類別
Metal.framework	8.0	MTL	動畫產製類別
MobileCoreServices.framework	3.0	UT	系統基本操作類別
MultipeerConnectivity.framework	7.0	MC	多點對點連結類別
NetworkExtension.framework	8.0	NE	VPN 應用類別

名稱	可用版本	前置碼	描述
NewsstandKit.framework	5.0	NK	電子書報類別
NotificationCenter.framework	8.0	NK	訊息通知類別
OpenAL.framework	2.0	AL	OpenAL 聲音處理類別
OpenGLES.framework	2.0	EAGL,GL	OpenGL ES 2D/3D 處理類別
PassKit.framework	6.0	PK	電子票券類別
Photos.framework	8.0	PH	圖片/影片處理類別
PhotosUI.framework	8.0	PH	圖片/影片處理 UI 類別
PushKit.framework	8.0	PK	VOIP 待接類別
QuartzCore.framework	2.0	CA	基本動畫類別
QuickLook.framework	4.0	QL	檔案預覽類別
SafariServices.framework	7.0	SS	Safari 存取類別
SceneKit.framework	8.0	SCN	3D 影像建立類別
Security.framework	2.0	CSSM,Sec	電子憑證安全類別
Social.framework	6.0	SL	社群處理類別
SpriteKit.framework	7.0	SK	動畫處理類別
StoreKit.framework	3.0	SK	InApp 購買類別
SystemConfiguration.framework	2.0	SC	系統裝置處理類別
Twitter.framework	5.0	TW	Twitter 存取類別
UIKit.framework	2.0	UI	系統介面類別
VideoToolbox.framework	6.0	VT	低階影片處理類別
WebKit.framework	8.0	WK	網頁內容整合類別

資料來源：developer.Apple.com

表9 著名第三方 API 程式庫

名稱	描述
ASIHTTPRequest	網路服務存取類別(HTTP request)
Facebook SDK	提供 Facebook 認證與服務
JSONKit/SBJSON	JSON 資料格式處理類別
KissXML/GDataXML/TBXML	XML 資料格式處理類別
plcrashreporter.org/Crashlytics	當機資訊回報
AFNetworking/Alamofire	網路存取服務(URL)
ZBar	一維條碼掃描 API

資料來源：本計畫整理

2.3.行動應用 App 資安風險議題

行動應用 App 的資安風險議題自 2008 年 Apple App Store 與 Google Play(前身為 Android 電子市場)兩大主流行動應用 App 商店登場以來至今從未停歇。

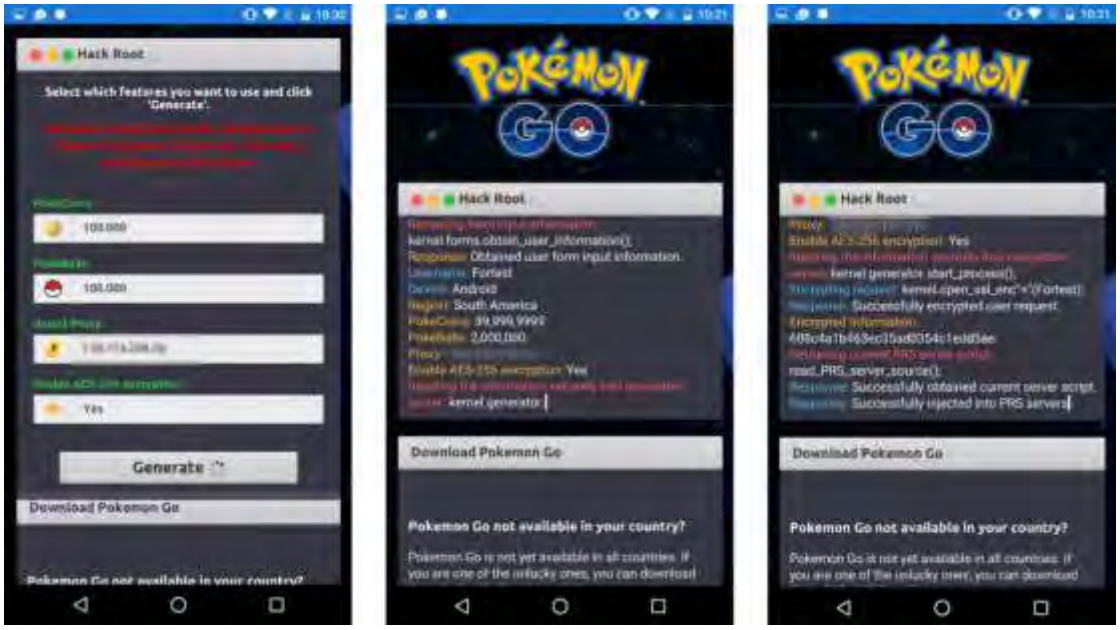
行動應用 App 商店提供了使用者與 App 開發人員(可能是中大型 App 開發商或是小型工作室或個人)，直接線上交易及下載 App 的市集，App 開發人員通過行動應用 App 商店上架免費或付費 App，使用者可以至商店下載，使得行動智慧裝置可以因裝置 App，將原來僅是提供打電話及收發簡訊的功能性手機進化成多功能智慧型手機。

為了在保障行動應用 App 的資訊安全，當 App 要上架至 Google Play 與 App Store，需要取得 App 憑證及接受行動應用平台的安全審核，確認沒有惡意，才能上架供使用者下載。

當然行動應用 App 也不一定要由前述 2 個商店下載，因為 Android 及 iOS 都有提供企業或政府組織可以建置自己的行動應用 App 市集，不用由 Google Play 或 App Store 審核，就可以發布企業或

政府組織內部使用的 App，惡意攻擊者也可能利用這個企業內行動應用 App 機制，將惡意 App 上架至偽冒行動應用 App 商店開放使用者下載。

以最近在世界各地引起旋風的以日本知名動畫《神奇寶貝》為基礎的手機遊戲「Pokémon Go」，在 2016/7/6 上線，在短短不到 1 個月，全球已下載達 1 億次以上，在此期間已出現一個夾帶木馬的冒牌版本，接著又出現鎖定螢幕程式，還會幫你偷偷點色情廣告的程式還有「Pokémon Go」攻略與破解安裝類的冒牌 App，暗藏著惡意的彈出視窗或廣告，會讓使用者意外訂購一些昂貴的非必要服務。或是宣稱使用者的手機感染了病毒必須加以清除，但事實上，使用者一旦點選了所提供的連結，就會幫你發送一封訂閱簡訊到某個昂貴的服務。



資料來源：趨勢科技，http://blog.trendmicro.com.tw/?cat=15 (105/8/11)

圖9　假圖帶有惡意程式「Pokémon Go」冒牌 App

除此之外，App 開發人員，常因限於開發時間緊迫、人力成本緊縮及未考慮到使用情境的安全問題，或是從未接受安全程式設計實務訓練，寫出有漏洞程式。

2.3.1. OWASP 十大行動安全風險 (Mobile Top 10 Risk)

國際知名開放軟體安全計畫(Open Web Application Security Project, OWASP)發布 2016 年行動應用 App 前 10 大安全風險候選清單及內容如下：

2.3.1.1. M1 不當使用行動作業平台(M1－Improper Platform Usage)

這個類別涵蓋了行動作業平台功能的失效或誤用平台的安全控制措施。可能包括 Android 的意圖(Intent)、作業系統平台的權限，或是 iOS 的 TouchID，Keychain 或其他一些行動作業系統內建安全控制的誤用。

2.3.1.2. M2 不安全資料儲存(M2-Insecure Data Storage)

這包括不安全的資料儲存和資料意外洩漏。

2.3.1.3. M3-不安全通訊(M3-Insecure Communication)

包括實作不良的交握過程，不正確 SSL 版本、弱的通訊交握協調及以明文傳遞敏感性資料。

2.3.1.4. M4-不安全身分認證(M4-Insecure Authentication)

此類別將涵蓋認證終端用戶或不當的交談管理(Session Management)的概念。這個風險包括：當使用者需要身分認證時認證機制失效、不能持續維持使用者身分認證及強度不足的交談管理。

2.3.1.5. M5-強度不足的加密(M5-Insufficient Cryptography)

程式碼加密運用於敏感資訊資產。然而，加密在某些方法實作強度不足。需要注意一切及任何有關 TLS 或 SSL 的加密的問題是屬於 M3。此外，如果行動應用 App 完全加密失效，這可能是屬於在 M2 的風險。該類別用於嘗試加密，但因實作不正確的問題。

2.3.1.6. M6-不安全授權(M6-Insecure Authorization)

這個類別是捕獲任何故障中的授權(例如：在客戶端的授權決定，強迫瀏覽等)的。不同於認證的問題(例如：設備註冊，用戶身分識別等)。如果 App 沒有在一般情況下，應該對用戶進行身分驗證時正常執行(例如：通過驗證，需要授權存取時給予一些資源或服務匿名存取)，那麼這是一個認證失敗不是一個授權失敗的問題。

2.3.1.7. M7-用戶端程式碼品質(M7-Client Code Quality)

這個類別是包含了所有的行動用戶端程式碼級的實作問題。這與從伺服器端的編碼錯誤是不同的。其中，解決方法是重新編寫的行動設備上執行的一些程式碼，這將可捕捉到緩衝區溢出、格式化字串漏洞，以及其他各種程式碼級的錯誤。

2.3.1.8. M8-程式碼篡改(M8-Code Tampering)

此類別包括二進制修補程式、本地資源修改、方法掛鉤(Method Hooking)、方法置換(Method Swizzling)和動態記憶體修改。一旦 App 被傳遞到行動設備，所述程式碼和資料資源駐留在那裡。攻擊者可以直接修改程式碼，動態改變記憶體中的內容，更改或替換 App 使用系統的 API 或修改 App 的資料和資源。這類軟體可以提供攻擊者竊取個人資料或獲取的預期用途的一個直接的方法。

2.3.1.9. M9-逆向工程(M9-Reverse Engineering)

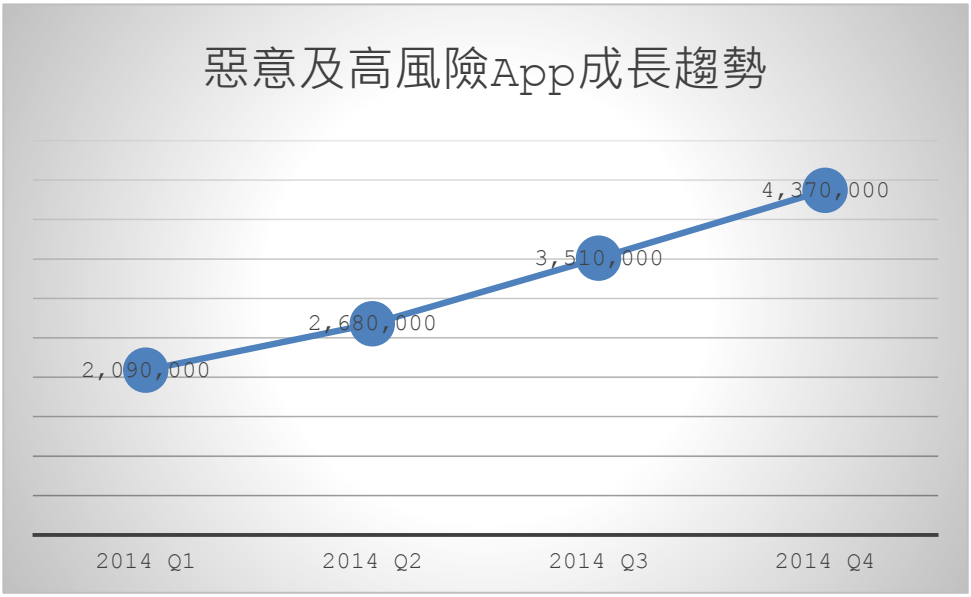
這一類包括行動應用 App 最終核心二進制原始程式碼、函式庫、演算法及其他資產的分析。軟體如 IDA Pro、Hopper，otool，和其他二進制檢測工具給攻擊者洞察 App 的內部工作。這可能被用來利用 App 中的其他新生的漏洞，以及有關後端伺服器，加密常量和密碼，以及智慧財產權揭示的資訊。

2.3.1.10. M10-多餘的功能(M10-Extraneous Functionality)

通常情況下，開發商包括隱藏的後門功能，或不打算被釋放到正式環境中的其他內部開發的安全控制。例如：開發人員可能會意外地包括一個密碼寫在混合應用註解中。另一個例子包括在測試過程中禁用雙因素身分驗證。

2.3.2. 惡意行動應用 App

除了上述開發人員無意造成行動應用 App 的漏洞外，有極大數量的行動應用 App 帶有惡意或隱密蒐集使用者敏感性資料的意圖，依趨勢科技全球技術支援與研發中心於 2014 年研究，2014 上半年，非重複性行動裝置惡意及高風險 App 程式樣本數量正式突破 200 萬大關，而且數量還在不斷增加，這距離上次突破 100 萬大關僅僅 6 個月的時間。此外，行動裝置惡意程式正逐漸演出成更精密的變種，例如：數位貨幣採礦惡意程式 ANDROIDOS_KAGECOIN 和行動裝置勒索程式 ANDROIDOS_LOCKER.A。到了 2014 下半年網路上惡意及高風險的 App 程式總數已達到 437 萬。這意味著，從該年上半年結束至今已成長了大約 68%，也就是 260 萬。



資料來源：趨勢科技

圖10 2014 年惡意及高風險的 App 程式成長趨勢-查詢時間：2016/10/1

其中 69%的行動裝置威脅都屬於越權廣告程式。越權廣告程式是指那些持續顯示廣告而讓使用者無法專心好好使用裝置的程式。其次是高費率服務盜用程式(PSA)，占惡意及高風險 App 程式的 24%左右。PSA 會在背景偷用一些高費率服務，讓使用者的手機帳單無故飆高。

2.3.3. 針對行動應用平台攻擊事件

不良的行動裝置使用習慣加上裝置本身的安全漏洞，將帶來重大的資安威脅。各平台都有類似的情形，其中最普遍的是 Android 和 iOS，這兩大平台的使用者都是惡意人士的熱門攻擊目標。

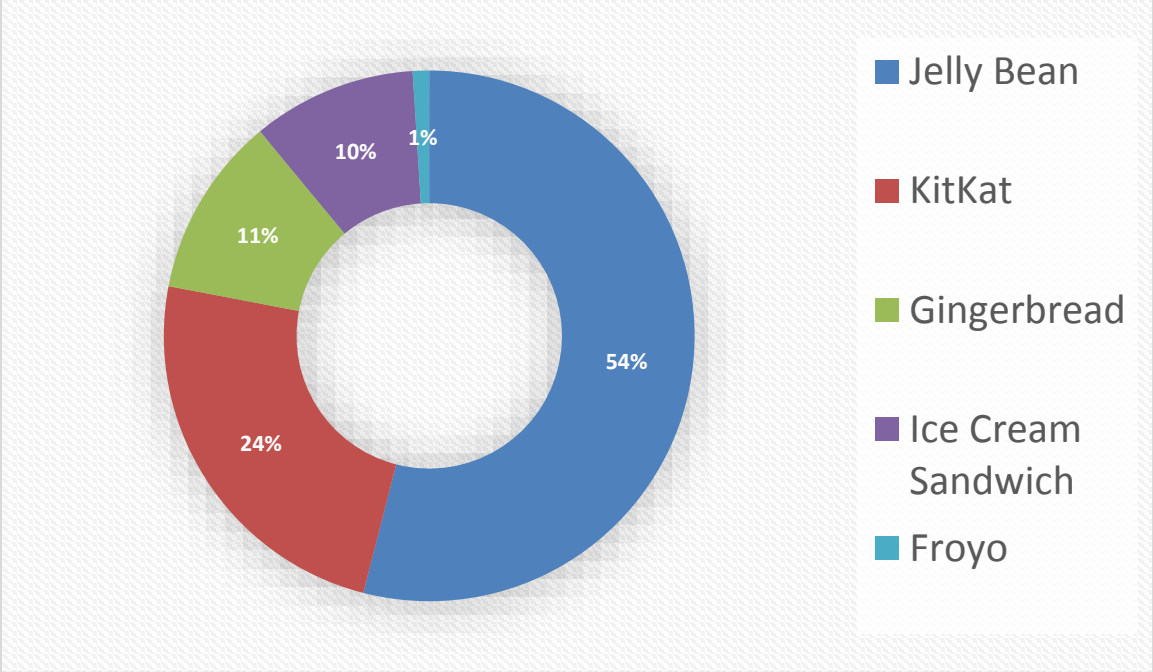
2.3.3.1. Android Fake ID 攻擊事件簿

Android 使用者一向是各種惡意和高風險 App 程式的目標。

Android 平台出現一個最新的漏洞，牽涉到交叉簽署憑證的處理機制。這項漏洞出現在 JarFile 和 KeyStore 這兩個 Android 架構經常用到的程式庫類別(class)。

除此之外，2014 下半年還有其他行動裝置漏洞造成使用者恐慌，例如：可讓惡意程式假冒正常程式的 Android FakeID 漏洞就引起了軒然版本的所有 Android 裝置(詳見圖 11 所示)都受到影響。

此外，還有 Android 內建瀏覽器讓駭客能夠避開「同源政策」(Same Origin Policy)的漏洞，這讓惡意程式可能存取正常網站所使用的資料和 Cookie。最近也發現網路犯罪集團會將 Facebook 使用者引導到專門利用此漏洞的惡意網站。



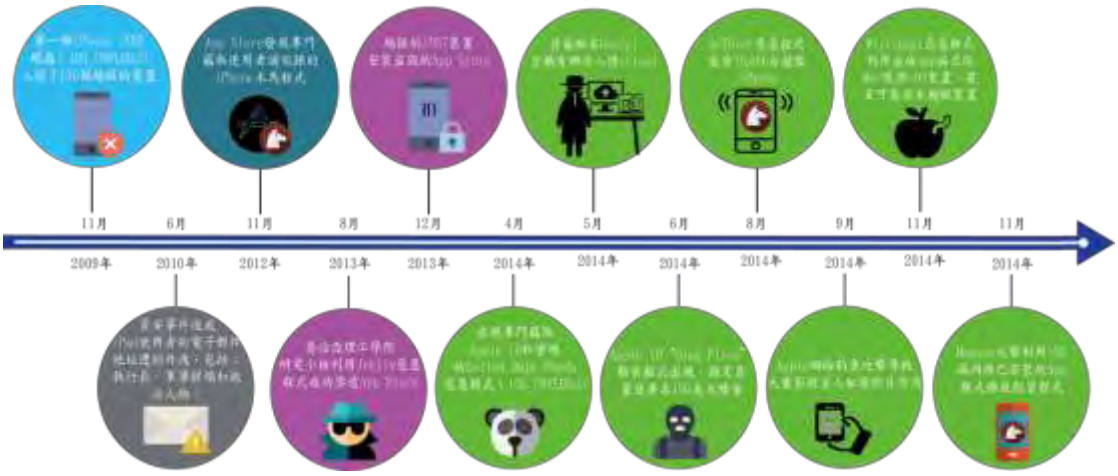
資料來源：趨勢科技

圖11 受 FakeID 及內建瀏覽器漏洞影響的 Android 作業系統版本

2.3.3.2. iOS 攻擊事件簿

而 iOS 方面，儘管 Apple 採取封閉式系統的作法來保護其平台，但網路上仍慢慢出現一些日益危險的威脅，成功利用了一些越獄所帶來的漏洞。

將iOS 資安事件自 2009 至 2014 時序列表詳見圖 12 所示。



資料來源：趨勢科技 查詢時間：2016/10/1

圖12 iOS 資安事件時序表(2009 至 2014 年)

2014 年 11 月，iOS 系統出現了兩個知名的威脅。

首先是 Mac 和 iOS 兩棲的 Wirelurker 惡意程式，由趨勢科技偵測到的 OSX_WIRELURK.A，是一個木馬化的 App 程式，可經由 USB 連線從 Mac 電腦感染 iOS 裝置。Wirelurker 甚至利用了一個偷來的憑證，能成功安裝在沒有越獄的裝置。僅管此惡意程式並未對裝置造成重大傷害，但這項攻擊技巧將促使網路犯罪集團開發出更具破壞力的 iOS 惡意程式。

其次是同一個月現身的 Masque 惡意程式，所有的 iOS 裝置(不論是否有越獄)都可能受害，惡意人士可能存取一些正常 App 程式所使用的非加密資料。Masque 和 Wirelurker 的攻擊，使得企業自行撰寫

及配發的 App 程式受到矚目，因為這已成為惡意人士藉機偷取個人資訊的管道。

由趨勢科技以上研究發現歸納，行動應用 App 常見威脅歸納如下：

- 行動裝置勒索程式：加密使用者資料，勒索使用者換取解密金鑰。
- 越權廣告程式：未經使用者同意取得或獲取過多權限，取用使用者敏感性資料。
- 費率服務盜用程式(PSA)：在背景偷用一些高費率服務，讓使用者的手機帳單費用無故飆高。
- 免越獄植入木馬：以竊取的憑證，能成功安裝在沒有越獄的裝置的木馬程式。
- 有漏洞程式庫/程式開發套件(SDK)：處理敏感性資料程式庫及開發工具漏洞被利用。
- 偽造的程式 ID(Fake ID):可讓惡意程式假冒正常程式的 Android FakeID 漏洞。
- 內建瀏覽器跨站來源來源腳本存取漏洞：惡意程式可能存取正常網站所使用的資料和 Cookie，取得敏感性資料。

此外，熱門 App 程式本身的漏洞，也為行動裝置威脅增添了更多攻擊管道。一些 App 內建支付機制程式開發套件(SDK)的漏洞，如 Spotify、Evernote 及支付寶等，讓使用者遭到了網路釣魚攻擊。

2.4.各國行動應用 App 安全開發要點簡介

智慧型手機使用者透過內建 Apple App Store、Google Play 等行動應用 App 市集，或是開發商的網頁，可以很輕鬆下載並安裝來自

世界各國開發人員所上架行動應用 App，相反地作為行動應用 App 開發商、小型工作室或獨立開發業人員，只要能依平台的開發指南進行開發並通過審核機制，就能上架市集到世界各國，對開發人員及使用者都很便利。

在 2015 年之前，就 2 大平台而言，Apple App Store 對上架 iOS App 的安全性的審查較 Google Play 的 Android App 來得嚴格。甚至 Google 允許開發人員自行發布 Android App，在數量在大幅超過 iOS App，但因沒有嚴格把關造成許多資訊安全漏洞或是蓄意蒐集使用者隱私、付費等敏感資料。Google 為了扭轉 Android 平台不安全形象，2015 年也轉為先審後上的機制。依新聞揭露得知 2 大主流平台使用工具及人工審查的時間都已縮短至 1~2 天，甚至 Google Play 可於數小時內完成。

雖然行動應用 App 上架均有平台採先審後上機制把關，各國政府的公權力亦不會強勢介入，為加強保護本國公民隱私及降低因行動應用 App 資訊安全風險，主流市場的歐盟、美國、中國大陸及日本都推出行動智慧裝置安全相關實務指引，供企業用戶、個人或開發人員參考，茲就有關各國有關安全開發實務重點摘要介紹如下(依英文字母順序)：

2.4.1. 中國大陸(China)

中國大陸為解決行動智慧裝置普及帶來惡意吸費、竊聽、竊錄及位置資訊洩露等安全事件頻繁的安全問題，遂由其工業與信息化部(以下簡稱工信部)，發布了一系列行動智慧裝置安全標準如下：

- 「行動智慧裝置安全能力設計導則」。
- YD/T 2407-2013「行動智慧裝置安全能力技術要求」。

●YD/T 2408-2013「行動智慧裝置安全能力測試方法」。

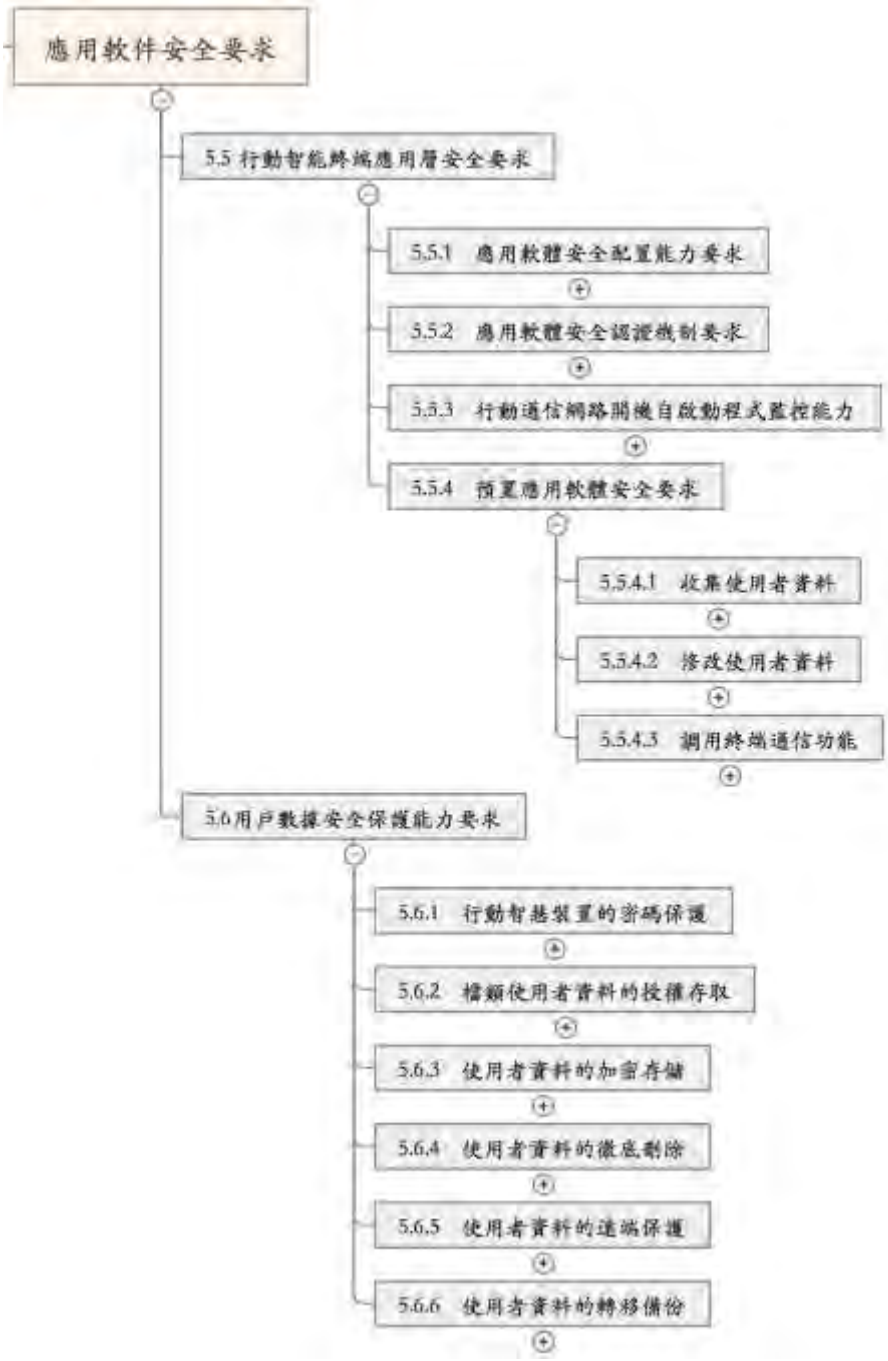
●YD/T 1886-2009「移動終端晶片安全技術要求和測試方法」。

其中 YD/T 2407-2013「行動智慧裝置安全能力技術要求」與 YD/T 2408-2013「行動智慧裝置安全能力測試方法」等 2 份標準規範，較接近行動應用 App 安全開發實務的要求，分別簡介如下：

2.4.1.1. YD/T 2407-2013「行動智慧裝置安全能力技術要求」

該標準於引言中一開始就揭示其基本原則為：「行動智慧裝置上發生的行為和應用要符合用戶的意願」，在此一原則規範下，行動智慧裝置不論是蒐集、儲存及呼叫個人資料在內之敏感資料與連網行為都需經使用者同意且符合預期。此標準為了有創新和發展，不規定具體的實現方法和措施，尚無法直接作為開發人員安全程式碼實作實務，但可作為安全開發準則參考之一。

此標準對行動智慧裝置的「硬體安全能力要求」、「作業系統安全能力要求」、「週邊接口安全能力要求」、「應用軟體安全要求」及「使用者資料安全保護能力要求」等 5 個層面提出要求，其中與行動應用 App 安全開發有關的條文為「5.5 行動智慧裝置應用層安全要求」與「5.6 行動智慧裝置使用者資料安全保護能力要求」等 2 部分，主要條文要求如下：



資料來源：YD/T 2407-2013

圖13 YD/T 2407-2013「行動智慧裝置安全能力技術要求」對應用層安全及用戶資料安全保護要求條文

此外,對行動智慧裝置安全能力自低到高劃分為 5 級，每一等級定義行動智慧裝置在相應等級對應的安全能力的最小組合，也就是行動

智慧裝置必須符合該組合中的所有安全能力才能標識為該等級，例如：達到第五級的行動智慧裝置應必須符合此標準第 5 章和第 6 章所定義的所有安全能力及功能限制性要求。具體有關應用層安全與保護使用者資料等級劃分詳見表 10 所示。

表10 行動智慧裝置安全能力分級(摘錄與行動應用 App 安全有關)

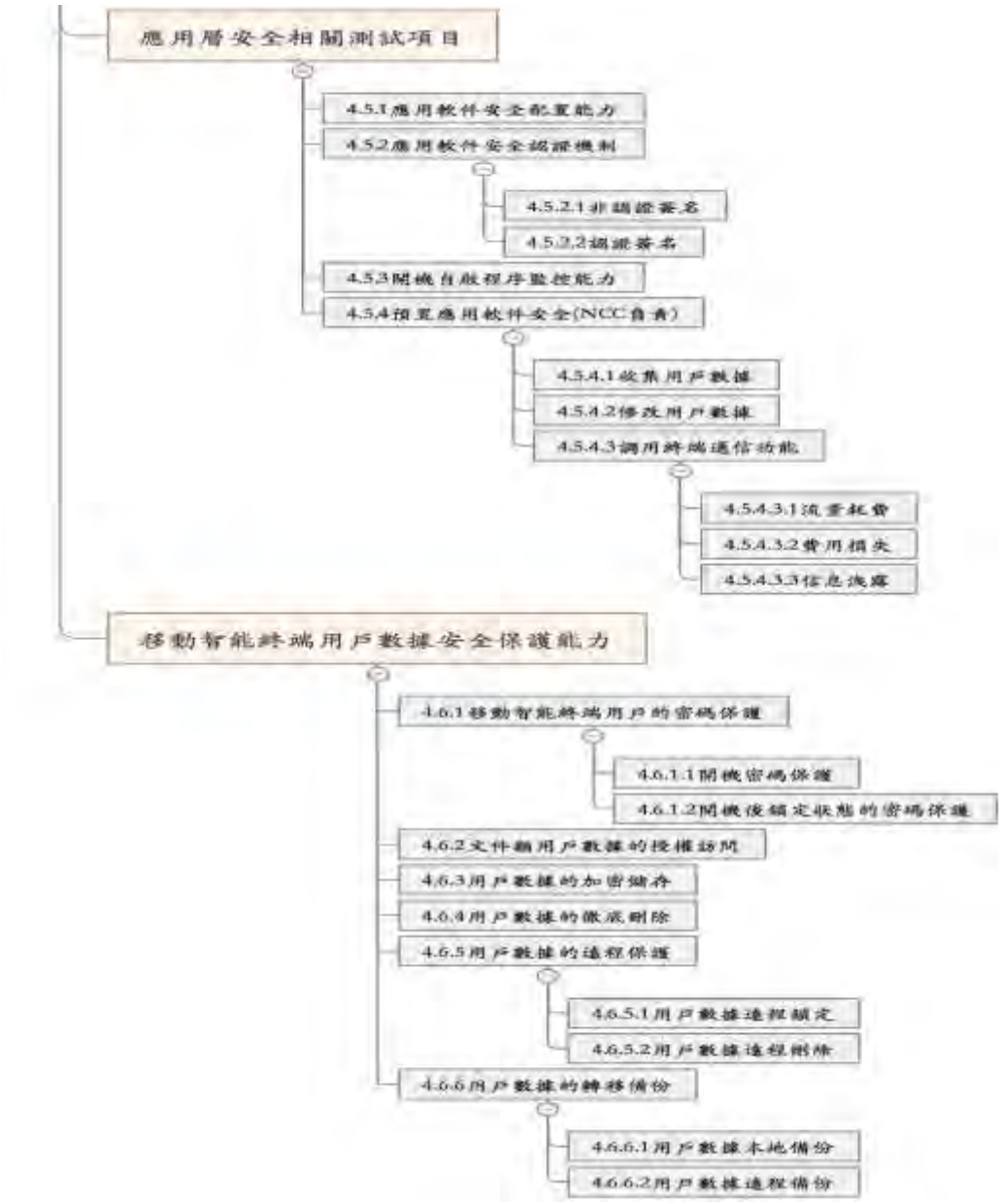
安全能力		安全能力等級				
		一級	二級	三級	四級	五級
28.	5.5.1 應用軟體安全配置能力要求				V	V
29.	5.5.2.1 非認證簽名要求	V	V	V	V	V
30.	5.5.2.2 認證簽名要求			V	V	V
31.	5.5.3 開機自啟動程式監控能力			V	V	V
32.	5.5.4.1 蒐集使用者資料	V	V	V	V	V
33.	5.5.4.2 修改使用者資料	V	V	V	V	V
34.	5.5.4.3.1 流量耗費	V	V	V	V	V
35.	5.5.4.3.2 費用損失	V	V	V	V	V
36.	5.5.4.3.3 資訊洩露	V	V	V	V	V
37.	5.6.1 行動智慧裝置的密碼保護	V	V	V	V	V
38.	5.6.2 檔類使用者資料的授權存取					V
39.	5.6.3 使用者資料的加密儲存					V
40.	5.6.4 使用者資料的徹底刪除			V	V	V
41.	5.6.5 使用者資料的遠端保護		V	V	V	V
42.	5.6.6 使用者資料的轉移備份		V	V	V	V

資料來源：YD/T 2407-2013

2.4.1.2. YD/T 2408-2013「行動智慧裝置安全能力測試方法」

此標準是 YD/T 2407-2013「行動智慧裝置安全能力技術要求」配套的測試方法。此標準針對技術要求提出的技術指標設計了相應且科學的測試方法，用於驗證行動智慧裝置是否滿足技術要求規定的內容。此標準可用於對行動智慧裝置安全能力的管理。通過此標準

可從測試方法角度保證行動智慧裝置安全能力要求的落地實施，切實地提高行動智慧裝置的安全能力。與行動應用 App 安全開發實務相關測試項目詳見圖 14 所示。



資料來源：YD/T 2408-2013

圖14 YD/T 2408-2013「行動智慧裝置安全能力測試方法」與行動應用 App 安全相關測試項目

●測試項目舉例一：4.5.2.2 認證簽名

測試編號：4.5.2.2
測試項目：認證簽名-應用程序安裝
專案要求：見 YD/T2407-2013 第 5.5.2.2 節
預置條件：被測移動智慧終端機處於正常工作狀態
測試步驟： 步驟 1：檢查被測移動智慧終端機是否提供應用軟體認證簽名機制； 步驟 2：對於提供了應用軟體認證簽名機制的移動智慧終端機，檢查移動智慧終端機是否支援 OCSP 或 CRL 協議線上檢查程式碼簽章憑證的有效性； 步驟 3：開發一款 App，該程式具有認證的程式碼簽名，簽名沒有被篡改過，且其簽章憑證處於有效期內；將該 App 安裝到被測智慧終端機上； 步驟 4：開發一款 App，該程式具有認證的程式碼簽名，簽章憑證處於有效期內，但其簽名被篡改過；將該 App 安裝到被測智慧終端機上； 步驟 5：開發一款 App，該程式具有認證的程式碼簽名，簽名沒有被篡改過，但其簽章憑證被吊銷；將該 App 安裝到被測智慧終端機上； 步驟 6：開發一款 App，該程式不具有認證的程式碼簽名。將該 App 安裝到被測智慧終端機上
預期結果： 在步驟 1 後，如果檢查到被測移動智慧終端機提供了應用軟體認證簽名機制，則繼續執行步驟 2； 在步驟 2 中，如果被測移動智慧終端機支援 OCSP 或 CRL 協議線上檢查程式碼簽章憑證的有效性，則繼續執 行步驟 3；否則，該專案評測結果為“不符合要求”，評測結束； 在步驟 3 中，App 應能夠成功安裝到被測智慧終端機上； 在步驟 4、5、6 中，如果被測移動智慧終端機禁止 App 安裝或提示使用者安裝的風險，則該專案評估結果為“未見異常”，否則該專案的評估結果為“不符合要求”

●測試項目舉例二：4.6.3 使用者資料的加密儲存

測試編號：4.6.3
測試項目：使用者資料的加密儲存
專案要求：見 YD/T2407-2013 第 5.6.3 節
預置條件：被測移動智慧終端機處於正常工作狀態
測試步驟： 步驟 1：將移動智慧終端機的電話本資料儲存於加密儲存區； 步驟 2：將加密後的電話本資料採用未授權方式匯出到其他設備(如本地電腦)上，用文本編輯軟體打開並檢視器是否以密文方式儲存
預期結果： 在步驟 2，使用文本編輯軟體應無法還原電話本檔 移動智慧終端機滿足以上預期結果，則該項目評測結果為“未見異常”，反之該專案評測結果為“不符合要求”

2.4.2. 歐盟(Europe Union,EU)

歐盟的行動應用 App 安全開發實務要點是由歐洲網路暨資訊安全局(European Network and Information Security Agency,ENISA)於 2011 年發布的「智慧型手機開發人員安全開發指引(Smartphone Secure Development Guidelines forAppDevelopers, SSDGAD)，SSDGAD 係與開源網頁 App 安全計畫基金會(Open Web Application Security Project Foundation,OWASP Foundation)行動安全專案計畫(Mobile Security Project)共同合作之計畫，是專為智慧手機 App 開發人員為導向，以開發安全的 App 為目標所編寫，同時也可以提供給行動應用 App 開發的專案經理參考。

SSDGAD 中的針對行動應用 App 的安全風險評估分別參考了 OWASP 3 個知名計畫的前 10 大風險：

●OWASP Mobile Top 10 Risks。

●OWASP Web Top 10 Risks。

●OWASP Cloud Top 10 Risks。

依上述風險組合 SSDGAD 共歸納出行動應用 App 前 10 大控制措施分類及 65 項控制措施項目，詳見表 11 所示。

表11 ENISA 智慧型手機開發人員安全開發指引前 10 大控制分類及實務項目數

控制措施分類	項目數
1.在行動設備上識別和保護敏感資料	14
2.安全地在設備上處理密碼憑據	10
3.確保敏感資料在傳輸過程中受保護	7
4.正確地建置用戶認證、授權和 Sessions 管理	6
5.保持後端的 API(服務)和平台(伺服器)的安全	5
6.保全與第三方服務和 App 資料整合	3
7.特別注意蒐集和同意的儲存蒐集和使用者資料	6
8.實行控制，以防止未經授權的存取付費的資源(錢包，手機簡訊，電話等)	7
9.確認安全發布/供應行動應用 App	3
10.小心檢查程式碼中的錯誤任何運行時解譯	4
合計	65

資料來源：ENISA

SSDGAD 在每個控制措施分類前面都描述了對應的風險，可以讓負責行動應用 App 開發專案的專案經理及開發人員，充分了解這些安全開發實務要解決何種安全議題，並以簡潔但不失明確的用字遣辭，提供開發人員安全開發實務的指引，對於較於複雜性的解決方案會附上由OWASP/Google/NIST/EU 及業界實務性的參考文獻，供開發人員進一步深入研究。

附錄 A 也附上一般程式設計最佳實務 13 條，如需要除了進行正常狀況下測試，也要額外進行誤用狀況下測試。

2.4.3. 日本(Japan)

日本唯一在公開資源找到是由日本智慧型手機安全論壇(Japan Smartphone Security Forum，JSSEC Forum)，於 2016 年 2 月所發布的 Android App 安全設計/安全程式碼編寫指南(Android Application Secure Design/Secure Coding Guidebook)，該指引的目的是促進 Android App 安全設計與安全程式碼編寫的綜合技巧，並保持及時更新與分享安全開發實務。

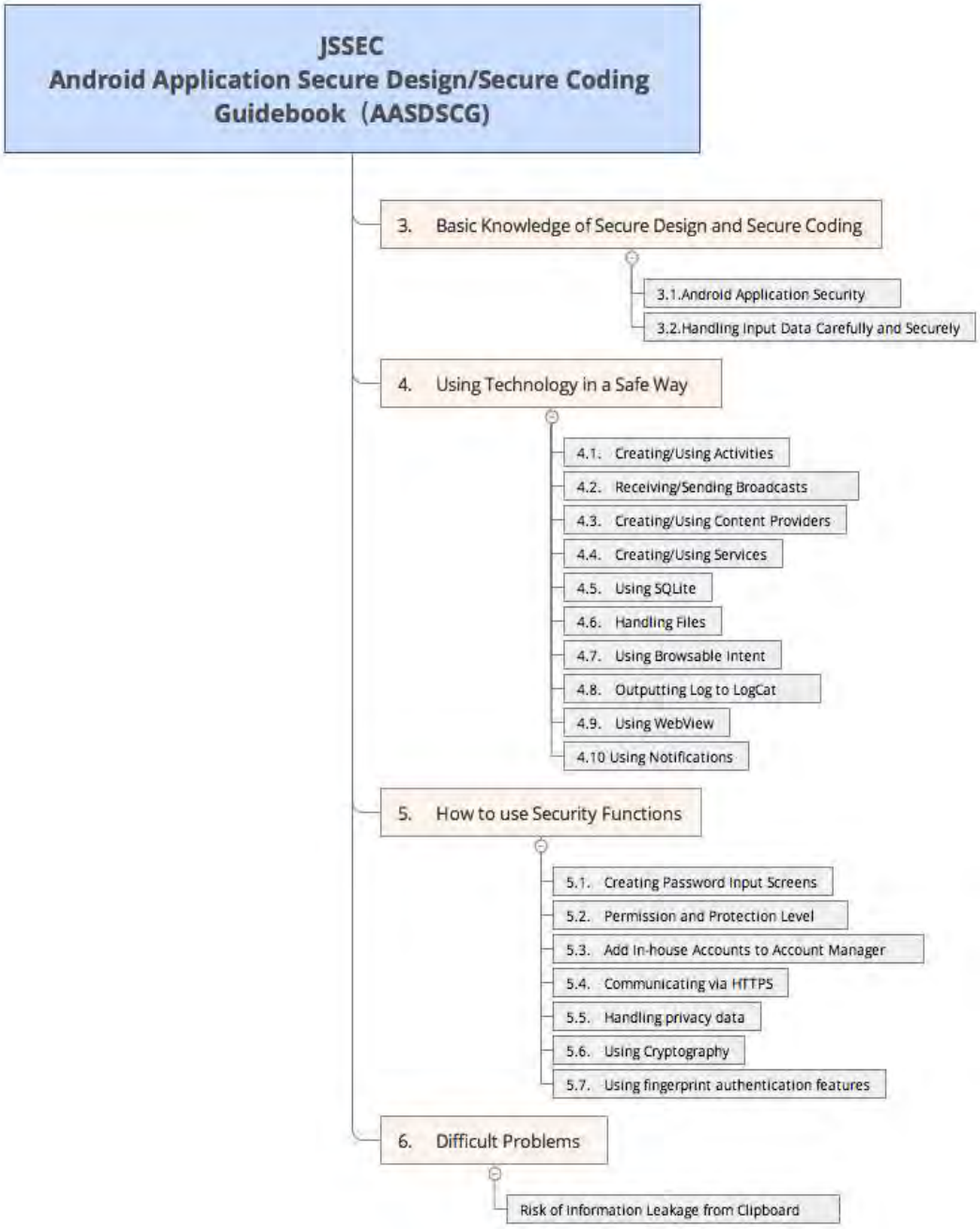
JSSEC 認為 Android 是開放式系統，需要較 iOS 花費更多精力去加強行動應用 App 安全性，因為 Android 可以不必透過 Google Play 來發布，缺乏了平台的安全把關，有較多被惡意運用存取，使用者敏感性資料或是有更多安全性漏洞可能被利用來進行惡意攻擊。

JSSEC 安全編碼小組，將盡最大努力保持包含在指南盡可能準確的內容，但 JSSEC 不保證完全無誤。JSSEC 僅保證上傳和宣傳最新版本當時那個特定的時刻最新，並接受開發人員回饋不同的方法的安全實務，採取發布定期 beta 版的方法，來維持更新。這個做法非常值得本指引後續維護參考。

最新版的指南與程式碼範例的 URL 如下：

- http://www.jssec.org/dl/android_securecoding_en.pdf Guidebook (English)(查詢日期:2016/10/1)
- http://www.jssec.org/dl/android_securecoding_en.zip Sample Codes (English)(查詢日期:2016/10/1)

JSSEC Android App 安全設計/安全程式碼編寫指南的技術指南架構(AASDSCG)詳見圖 15 所示。



資料來源：JSSEC

圖15 JSSEC Android App 安全設計/安全程式碼編寫指南的技術指南架構(AASDSCG)

2.4.4. 美國(United States of America, USA)

美國與行動應用 App 安全開發實務最為貼近應為由美國商務部所屬國家標準與科技機構(National Institute of Standards and Technology,U.S. Department of Commerce,NIST,DoC)所發布的 NIST Special Publication 800-163 行動應用 App 安全審核(Vetting the Security of Mobile Applications)這份國家標準文件。

此文件為了幫助企業的目的如下：

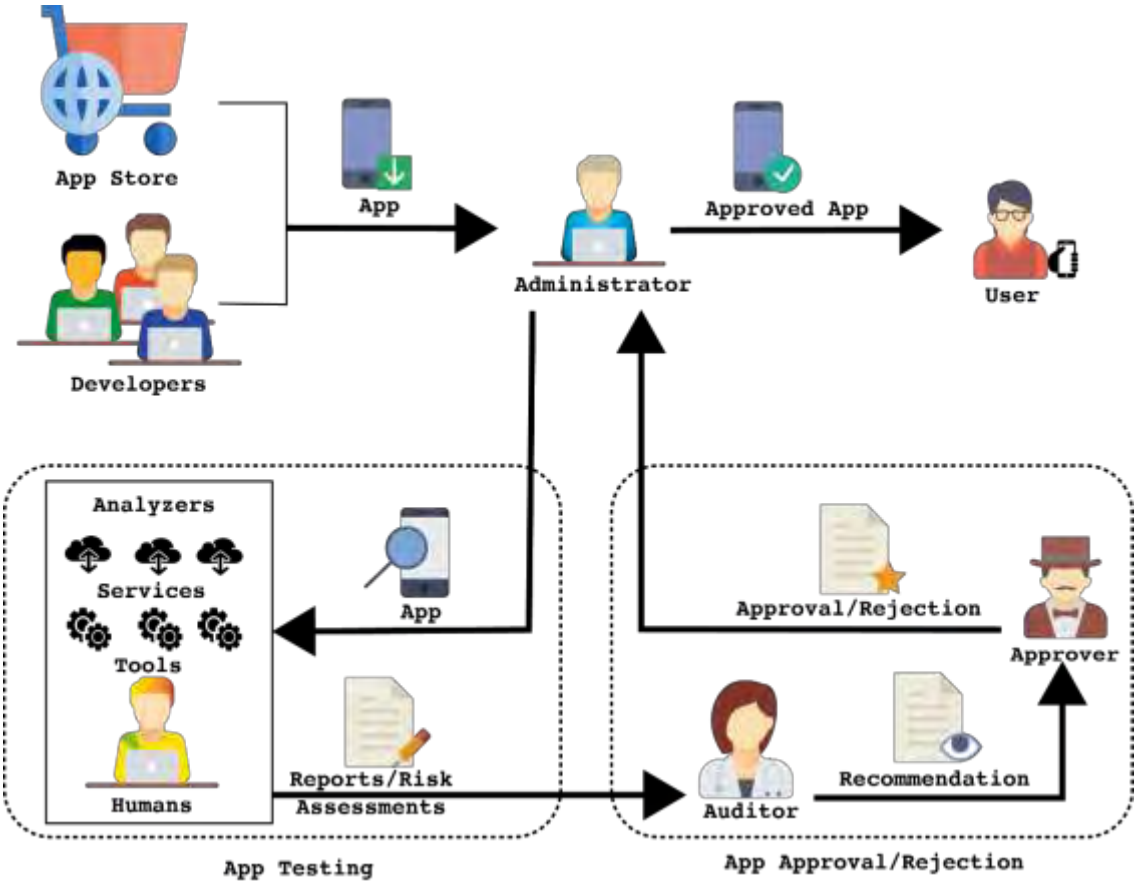
- 了解 App 的審核流程。
- 規劃實施應用審核流程。
- 開發行動應用 App 的安全性要求。
- 了解各類行動應用 App 的漏洞和測試用於檢測這些漏洞的方法。
- 確定一個行動應用 App 是在單位的行動智慧裝置部署可以接受的。

最終，行動應用 App 的接受取決於組織的安全要求，反過來，取決於其行動應用 App 部署環境，背景行為將在其中被使用，並用於運作行動應用 App 底層的行動技術。此文件強調那些行動應用 App 被批准之前，被認為是特別重要的內容適合直接使用的。

適用對象為計劃自行建置行動應用 App 審核流程；或利用其他組織現有的行動應用 App 的審核結果；也適用於開發有興趣了解，可以在自己的行動應用 App 的軟體開發生命週期中出現的軟體漏洞。

在 NIST SP 800-163 文件第 2 節中，以企業的觀點將由行動智慧裝置平台市集中，購買或由企業自行開發的行動應用 App 的安全性，

透過以不同權責角色分工，由行動應用 App 管理者(Administrator)，送到安全分析師(Security Analyzer)以自動化及人工方式測試安全，產出風險評估報告，送交稽核人員覆核並提出建議，最後由權責主管核准或駁回，最後才能決定是否發布給企業員工使用，建議流程圖詳見圖 16 所示。



資料來源：NIST SP 800-163

圖16 行動應用 App 審核流程及角色

NIST SP 800-163 在第 3 節中說明行動應用 App 測試活動說明，包括 3.1 一般安全性要求、3.2 的測試方法及 3.3 分享結果等 3 部分，其中 3.1 一般安全性要求可作為安全開發實務的參考；3.2 的

測試方法可作為行動應用 App 檢測漏洞分析參考；3.3 節分享結果。

一個組織的行動應用 App 調查結果共享，可以大大減少其他組織應用審查工作的重複和成本。如在資訊軟體品質保證社群內共享，可獲得來自世界各地的安全專家的集體努力協助分析的益處。美國國家漏洞資料庫(NVD)是採用基於標準的漏洞管理流程的美國政府資料庫，以使用安全內容自動化協議(SCAP)來呈現。此文件也提及如果企業採取其他組織的測試報告，可以跳過這個步驟。

第 4 章介紹行動應用 App 核准及駁回，說明稽核員如何執行風險稽核及組織定義的審核標準。

NIST SP 800-163 的安全實務要求描述非常清楚但是較為冗長，需要由段落中分析萃取出安全開發實務準則。

2.4.5. 雲端安全聯盟的行動應用 App 安全測試倡議白皮書

雲端安全聯盟(Cloud Security Alliance, CSA)的行動應用 App 安全測試(Mobile Application Security Testing, MAST)的目標是定義一個框架，落實安全的行動應用 App 的開發、設計實現隱私和安全。MAST 的實施將導致明確闡述建議，並在使用行動應用 App 的最佳實踐。

通過利用 MAST 行動應用 App 安全測試和審核過程涉及的靜態和動態分析，以評估行動應用 App 的安全漏洞的平台，如：Android、iOS 及 Windows。這些過程涵蓋權限、暴露通訊、潛在的惡意功能、應用勾結、混淆是非、過度的功耗及傳統的軟體漏洞。測試和審查程式也將涵蓋內部通訊，如：除錯標誌和活動，以及外部通訊，例如：全球定位系統(GPS)、藍芽、近場通訊(NFC)及全球行動通訊系

統(GSM)存取。除了行動應用 App 安全測試和審核，行動應用 App 的安全事件應變計畫也將得到發展。

MAST 引用 NIST SP 800-163 作為對價的識別基本的安全檢查規範分類等級的基礎。這些基準提供第三方機構提供相關的應用安全檢查，分析審核結果，評估安全風險為行動應用 App 和相應的安全等級評級，從而大大增強了行動應用 App 的安全性。此外，該白皮書概述了行動應用的必要的安全審查要求和基線。MAST 的章節架構詳見圖 17 所示。

資料來源：Cloud Security Alliance

圖17 CSA MAST 章節架構

CSA MAST 帶入行動應用 App 為角度的安全管理的生命週期，由開發(Development)、測試(Testing)、產品上架(Production)、更新(Update)、App 移除(App Removal)及 App 資料刪除(App Data Deletion)等階段實務，更參考 NIST SP 800-163 將行動應用 App 在生命週期各階段安全要求、App 測試及審核方法，以更結構化方式呈現。

2.4.6. 各國行動應用 App 開發實務小結

檢視中國大陸、歐盟、美國、日本及雲端安全聯盟行動應用 App 安全實務後，進行分析與萃取實務精華超過 250 條目，與我國「行動應用 App 基本資安規範」(2.5 節)及「行動應用 App 基本資安檢測基準」(2.6 節)進行比對、歸納至第 3 章行動應用 App 安全開發最佳實務。

2.5.我國「行動應用 App 基本資安規範」簡介

我國「行動應用 App 基本資安規範」(以下簡稱 App 資安規範)為 104 年由經濟部工業局委託財團法人資訊工業策進會(以下簡稱資策會)依據 103/6/24 行政院國家資通安全會報第 26 次委員會議決議，積極研議行動應用 App 基本資安規範。

資策會邀集國內資安領域專家成立工作小組，參照國際相關資安規範，歷經多次工作小組會商、專家座談研討等會議，蒐集產官學研先進之建議，並公開徵詢各界意見完成 App 資安規範，供業界開發行動應用 App 自主遵循參考。

App 資安規範係屬非強制性規定，主要目的在於提升我國行動應用 App 基本安全防護能力，從設計初始階段即導入基本資安概念，透

過規範之重點要項，提醒 App 開發人員強化資訊安全意識，並逐步完善自身 App 安全防護能力。

本規範從「行動 App 發布安全」、「敏感性資料保護」、「付費資源控管安全」、「行動 App 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」等 5 個層面提出資訊安全技術要求，App 開發人員可參考規範，自主提升所開發之行動應用 App 安全品質，增進使用者之信賴度與使用意願，創造 App 開發商與使用者雙贏局面。

主要技術要求在第 4 章，區分為 4.1 行動 App 資訊安全技術要求事項及 4.2 伺服端資訊安全技術要求事項等 2 節，置重點在 4.1 節行動應用 App 端的安全技術要求事項，共有 17 項技術要求事項，4.2 節僅簡單建議應由業者自我宣告或切結其伺服端資訊安全防護與管理措施，或對於其伺服端服務之資訊安全防護與管理，出具第三方檢測通過證明。

此外針對 App 應用類別在第 5 章提出不同安全性要求，共分為 3 類，分別為：

- 第一類：純功能性。
- 第二類：具認證功能與連網行為。
- 第三類：具交易功能(包括認證功能及連網行為)。

針對每一安全分類，定義應符合資訊安全技術要求事項之最小集合，即 App 應符合其所屬分類中之所有資訊安全技術要求事項，非屬上述分類之特殊情況，於檢測標準另行說明。各安全分類之資訊安全技術要求事項詳見表 12 所示。

表12 各安全分類之資訊安全技術要求事項

編號	資訊安全技術要求事項	安全分類		
		一	二	三
1	4.1.1.1.行動 App 發布	V	V	V
2	4.1.1.2.行動 App 更新	V	V	V
3	4.1.1.3.行動 App 安全性問題回報	V	V	V
4	4.1.2.1.敏感性資料蒐集	V	V	V
5	4.1.2.2.敏感性資料利用	V	V	V
6	4.1.2.3.敏感性資料儲存	V	V	V
7	4.1.2.4.敏感性資料傳輸		V	V
8	4.1.2.5.敏感性資料分享	V	V	V
9	4.1.2.6.敏感性資料刪除	V	V	V
10	4.1.3.1.付費資源使用			V
11	4.1.3.2.付費資源控管			V
12	4.1.4.1.使用者身分認證與授權		V	V
13	4.1.4.2.連線管理機制		V	V
14	4.1.5.1.防範惡意程式碼與避免資訊安全漏洞	V	V	V
15	4.1.5.2.行動 App 完整性			V
16	4.1.5.3.函式庫引用安全	V	V	V
17	4.1.5.4.使用者輸入驗證	V	V	V

資料來源：經濟部工業局行動應用 App 基本資安規範

此 App 資安規範也參考了 2.4 節的各國規範(CSA MAST 為 2016 年 6 月發布除外)，也參考了我國「個人資料保護法」及 ISO/IEC 27001:2013 等多項國際資訊安全相關標準，在性質上屬於技術中立的高階行動應用 App 資安管理規範，為協助各採用組織及開發機構導循落實，經濟部工業局目前最新發布「行動應用 App 基本資安檢測基準 V2.0」，本指引將於 2.6 節進一步簡介。

2.6.我國「行動應用 App 基本資安檢測基準」及自主檢驗制度簡介

2.6.1.「行動應用 App 基本資安檢測基準」簡介

由於行動智慧裝置之普及，App 的使用儼然已成為國人的部分生活習慣。但考量開發人員可能在資訊安全意識或技術上之缺乏，使得 App 造成使用者之個人資料外洩或財務損失之風險。因此考量行動應用 App 之安全性，行動應用 App 基本資安檢測基準(以下簡稱檢測基準)應運而生。

檢測基準主要依據「行動應用 App 基本資安規範」中對行動應用 App 的安全分類，且參考 OWASP(開放 Web 軟體計畫)之「Mobile Security Project-Top Ten Mobile Risks 與 NIST(美國國家標準技術研究所)之「Special Publication 800-163 Vetting the Security of Mobile Applications」，評估及審驗相關行動應用 App 之風險項目，藉此制定安全檢測項目並規劃提出各項目之細項檢查事項、執行條件及預期結果等。本檢測基準可作為第三方檢測機構進行檢測之參考基準，而也可作為開發人員針對 App 安全進行檢測之依據。

本檢測基準適用於「行動應用 App 基本資安規範」中提出之所有行動應用 App 的安全分類，且不限於特定開發應用領域。本檢測基準為提供開發人員提升 App 安全水準之參考，並提出可能造成資安風險之項目以利開發人員進行檢測，並期望透過此基本資安檢測基準可提升開發人員在資訊安全上之意識素養與技術能力。

在此檢測基準提出 5 大類行動應用 App 檢測基準，分別為「行動應用 App 發布安全」、「敏感性資料保護」、「付費資源控管安

全」、「身分認證、授權與連線管理安全」及「行動應用 App 碼安全」並細分為 17 項之檢測項目做為控制項目。

此外，考量行動應用 App 服務後端伺服器安全，檢測基準也提供「伺服端安全管理」與「伺服端安全檢測」之項目內容，提供相關可供安全開發技術參考。

資料來源：經濟部工業局行動應用 App 安全基本規範

圖18 經濟部工業局行動應用 App 安全基本規範框架

在資安檢測基準中相關的檢測項目欄位說明詳見表 13 所示。

表13 行動應用 App 基本資安檢測基準欄位彙整表

欄位名稱	欄位說明
檢測編號	依據「行動應用 App 基本資安規範」之「4.技術要求」編號項次，檢測編號由 5 碼組成，分別為 4.1.x.y.z，「4.1.」表示為「行動 App 基本資安檢測基準」，「x.y.z」分別為其向下所展開之次編號項目。

欄位名稱	欄位說明
檢測項目	參照「行動應用 App 基本資安規範」之「4.技術要求」內容，訂定檢測摘要簡稱
檢測分級	▪「初級」檢測項目： 主要檢測無連網之基礎功能安全性，檢測方式採自動化工具為主 ▪「中級」檢測項目(含初級)： 主要檢測連網及認證安全性，檢測方式採人工檢測方式為主 ▪「高級」檢測項目(含中級)： 主要檢測付費資源安全性，檢測方式採人工檢測方式為主
檢測依據	依據「行動應用 App 基本資安規範」之「4.技術要求」相對應之行動 App 資訊安全技術要求事項技術要求
技術要求	依據「行動應用 App 基本資安規範」之「4.技術要求」相對應之行動 App 資訊安全技術要求事項「內容」
檢測基準	依檢測項目所須檢測之各項檢查事項
檢測結果	依據檢查事項，預期之檢測結果及各結果之形成條件。預期之檢測結果包括「符合要求」與「不符合要求」
備註	其他說明事項

資料來源：經濟部工業局行動應用 App 基本資安規範

資安檢測基準針對行動應用 App 提出 5 大項共 42 個檢測項目，彙整各檢測項目之編號名稱與技術要求詳見表 14 所示。

表14 行動應用 App 基本資安檢測基準項目簡表

檢測編號	檢測項目	技術要求
4.1.1.1.1	行動應用 App 應於可信任來源之行動 App 商店發布	建議參考項目
4.1.1.1.2	行動應用 App 發布說明	行動應用 App 應於發布時說明欲存取之敏感性資料、行動裝置資源及宣告之權限用途
4.1.1.2.1	行動應用 App 應於可信任來源	建議參考項目

檢測編號	檢測項目	技術要求
	之行動 App 商店發布更新	
4.1.1.2.2	行動應用 App 應提供更新機制	建議參考項目
4.1.1.2.3	行動應用 App 應於安全性更新時主動公告	建議參考項目
4.1.1.3.1	行動應用 App 問題回報	行動應用 App 開發人員應提供回報安全性問題之管道
4.1.2.1.1	行動應用 App 敏感性資料蒐集聲明	行動應用 App 應於蒐集敏感性資料前，取得使用者同意
4.1.2.1.2	行動應用 App 提供使用者拒絕敏感性資料蒐集機制	行動應用 App 提供使用者拒絕敏感性資料蒐集機制
4.1.2.2.1	行動應用 App 應於使用敏感性資料前，取得使用者同意	建議參考項目
4.1.2.2.2	行動應用 App 應提供使用者拒絕使用敏感性資料之權利	建議參考項目
4.1.2.2.3	行動應用 App 如採用通行碼認證，應主動提醒使用者設定較複雜之通行碼	建議參考項目
4.1.2.2.4	行動應用 App 應提醒使用者定期更改通行碼	建議參考項目
4.1.2.3.1	行動應用 App 敏感性資料儲存聲明	行動應用 App 應於儲存敏感性資料前，取得使用者同意
4.1.2.3.2	行動應用 App 提供使用者拒絕敏感性資料儲存機制	行動應用 App 應提供使用者拒絕儲存敏感性資料之權利
4.1.2.3.3	行動應用 App 儲存之敏感性資料，應僅用於其使用聲明之用途	建議參考項目
4.1.2.3.4	行動應用 App 敏感性資料儲存限制	行動應用 App 應避免將敏感性資料儲存於暫存檔或紀錄檔中
4.1.2.3.5	行動 App 敏感性資料儲存保護	敏感性資料應採用適當且有效之金鑰長度與加密演算法，進行加密處理再儲存

檢測編號	檢測項目	技術要求
4.1.2.3.6	行動應用 App 敏感性資料儲存控管	敏感性資料應儲存於受作業系統保護之區域，以防止其他 App 未經授權之存取
4.1.2.3.7	行動應用 App 敏感性資料硬碼(Hard Code)	敏感性資料應避免出現於行動應用 App 之程式碼
4.1.2.4.1	行動應用 App 敏感性資料傳輸	行動應用 App 透過網路傳輸敏感性資料，應使用適當且有效之金鑰長度與加密演算法進行安全加密
4.1.2.5.1	行動應用 App 敏感性資料分享聲明	行動裝置內之不同行動應用 App 間，應於分享敏感性資料前，取得使用者同意
4.1.2.5.2	行動應用 App 提供使用者拒絕敏感性資料分享機制	行動應用 App 應提供使用者拒絕分享敏感性資料之權利
4.1.2.5.3	行動應用 App 敏感性資料分享權限控管	行動應用 App 分享敏感性資料時，應避免未授權之行動 App 存取。
4.1.2.6.1	行動應用 App 如涉及儲存使用者敏感性資料，應提供使用者刪除之功能	建議參考項目
4.1.3.1.1	行動應用 App 付費資源使用聲明	行動應用 App 應於使用付費資源前主動通知使用者
4.1.3.1.2	行動應用 App 拒絕付費資源使用機制	行動應用 App 應提供使用者拒絕使用付費資源之權利
4.1.3.2.1	行動應用 App 付費資源使用者認證	行動應用 App 應於使用付費資源前進行使用者認證
4.1.3.2.2	行動應用 AppApp 付費資源紀錄	行動應用 AppApp 應記錄使用之付費資源與時間
4.1.4.1.1	行動應用 App 使用者身分認證機制	行動應用 App 應有適當之身分認證機制，確認使用者身分

檢測編號	檢測項目	技術要求
4.1.4.1.2	行動應用 App 使用者身分授權	行動應用 App 應依使用者身分授權
4.1.4.2.1	行動應用 App 交談識別碼規則性	行動應用 App 應避免使用具有規則性之交談識別碼
4.1.4.2.2	行動應用 App 伺服器憑證有效性	行動應用 App 應確認伺服器憑證之有效性
4.1.4.2.3	行動應用 App 伺服器憑證簽發來源	行動應用 App 應確認伺服器憑證為可信任之憑證機構、政府機關或企業簽發
4.1.4.2.4	行動應用 App 連線安全	行動應用 App 應避免與未具有效憑證之伺服器，進行連線與傳輸資料
4.1.5.1.1	行動應用 App 惡意程式碼	行動應用 App 應避免含有惡意程式碼
4.1.5.1.2	行動應用 App 資訊安全漏洞	行動應用 App 應避免資訊安全漏洞
4.1.5.2.1	行動應用 App 應使用適當且有效之完整性驗證機制，以確保其完整性。	建議參考項目
4.1.5.3.1	行動應用 App 函式庫引用安全	行動應用 App 於引用之函式庫有更新時，應備妥對應之更新版本，更新方式請參酌 4.1.1. 行動 App 發布安全
4.1.5.4.1	行動應用 App 使用者輸入檢查	行動應用 App 應針對使用者輸入之字串，進行安全檢查
4.1.5.4.2	行動應用 App 注入攻擊防護機制	行動應用 App 應提供相關注入攻擊防護機制

資料來源：本計畫整理

2.6.2.「行動應用 App 基本資安自主檢測推動制度」簡介

在部分 App 開發人員缺乏資安意識之情況下，將導致使用者面臨資料外洩或財產損害。前章所述之「行動應用 App 基本資安檢測基準」可作為開發人員在強化行動應用 App 之參考基礎，而經濟部工業局為落實 104 年 4 月公告之「行動應用 App 基本資安規範」，續專案計畫委託財團法人資訊工業策進會制定本規章，作為推動我國行動應用 App 自主檢測制度發展之依據。自主檢測推動制度重點說明以下：

2.6.2.1. 第一部分：行動應用 App 基本資安自主檢測推動制度規章

- 制度目的。
- 適用範圍。
- 定義。
- 自主檢測體系。
- 制度推動委員會。
- 認證機構。
- 檢測實驗室。
- 行動應用 App 基本資安標章(MAS 標章) 。
- 資訊控制。
- 追蹤管理。
- 費用。

2.6.2.2. 第二部分：行動應用 App 基本資安檢測實驗室資格認證及管理規範

●基本原則。

●檢測實驗室認可程序審查。

●補正期間。

●檢測實驗室認證證書。

●檢測實驗室人員守密原則。

●檢測實驗室費用原則。

●檢測實驗室之權利義務。

2.6.2.3. 第三部分：行動 App 基本資安標章使用與管理規範

●基本原則。

●名詞定義。

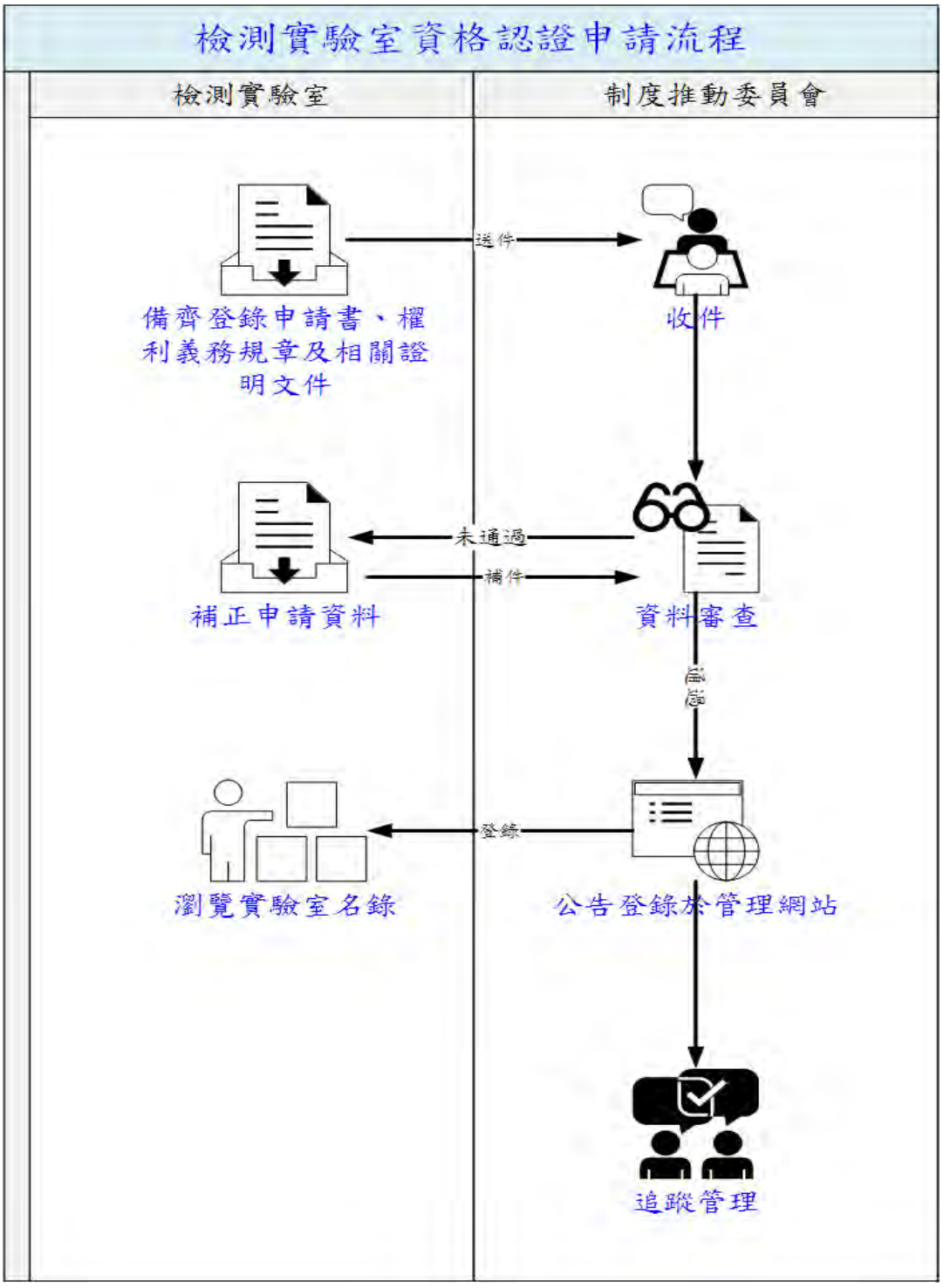
●標章之核發與使用。

●標章之更新與資訊通知。

●標章之追蹤管理。

●費用。

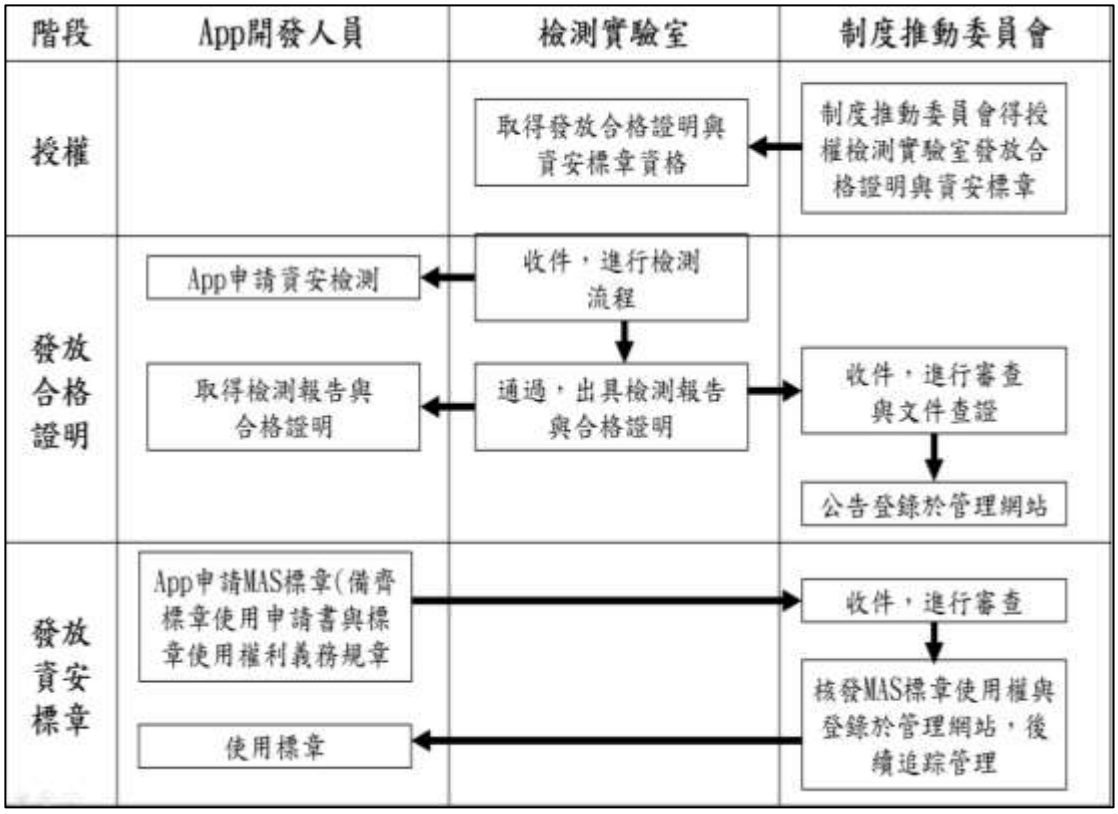
推動制度提供檢測實驗室提出檢測資格認證申請，資格認證申請流程如下：



資料來源：經濟部工業局行動應用 App 基本資安自主檢測推動制度 V3.0

圖19 檢測實驗室資格認證申請流程

為提供 App 開發人員可取得 MAS 標章宣告其 App 之資安檢測驗證，以下說明行動應用 App 開發人員申請檢測與 MAS 標章流程：



資料來源：經濟部工業局行動應用 App 基本資安自主檢測推動制度 V3.0

圖20 行動應用 App 開發人員申請檢測與 MAS 標章流程

3. 行動應用 App 安全開發最佳實務

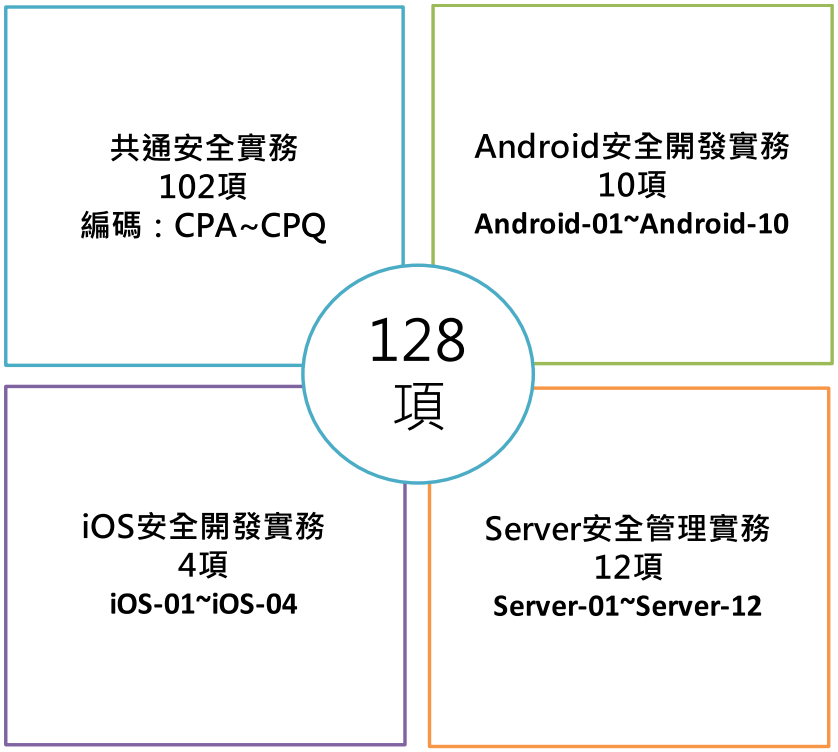
本指引於第 2.4 節簡介中國大陸、歐盟、日本、美國及 CSA 行動應用 App 安全安全開發實務要點，並依據我國經濟部工業局「行動應用 App 基本資安規範」(2.5 節)及「行動應用 App 基本資安檢測基準」(2.6 節)，本章歸納整理前述安全開發實務、規範及基準，共區分為 4 個實務類別共 128 項實務並進行編碼，詳見圖 21 所示及說明如下：

3.1 共通實務(Common Practices, CPA01~CPQ10 共 102 項)

3.2 Android 實務(Android-01~Android-10 共 10 項)

3.3 iOS 實務(iOS-01~iOS-04 共 4 項)

3.4 Server 實務(Server-01~Server-12 共 12 項)



資料來源：本計畫整理

圖21 本指引歸納整理區分 4 個實務類別/共 128 項實務準則

部分實務提供不安全程式碼或安全程式碼範例做參考。每項實務結構如下：

●**實務類別(CP|Android| iOS |Server)-編碼**：實務要點主題敘述。

●**說明**：

說明該項實務因應行動應用 App 或伺服器端安全風險及對應共通、Android、iOS 或 Server 端安全開發實務要點或技術解決方案。

●**安全開發生命週期**

說明該實務運用於安全開發生命週期單一或多個階段(需求、設計、開發實作、測試或部署維運階段)。

●**不安全程式碼範例**

提供 Android/iOS 與該實務相關不安全程式碼範例，程式碼的範圍以粉紅網底標示。

Android
iOS
網站伺服器

●**安全程式碼範例**

提供 Android/iOS 與該實務相關安全程式碼範例。程式碼的範圍以淺綠網底標示。

Android
iOS
網站伺服器

●**行動應用 App 基本資安規範**

說明該實務對應經濟部工業局「行動應用 App 基本資安規範的要求事項，如「4.1.2.2. 敏感性資料利用」。

●**行動應用 App 基本資安檢測基準**

說明該實務對應經濟部工業局「行動應用 App 基本資安檢測基準的要求事項，如「4.1.4.2.3. 如行動應用程式使用安全加密傳輸協定，檢查行動應用程式是否驗證並確保伺服器憑證為行動作業系統內建可信任之憑證機構、政府機關、企業簽發。」

●**各國開發要點參考**

說明本實務參考國際行動應用 App 安全開發要點來源，如 NIST SP 800-163 3.1.1

●**參考來源**：

說明該實務 Android、iOS 及網站伺服器等安全開發實務參考來源。

為使讀者方便查找到需要參考安全開發實務，本指引建立對應表如下：

- 依經濟部工業局「行動應用 App 基本資安規範」資訊安全技術要求事項對應詳見表 15 所示。
- 以「行動應用 App 基本資安檢測基準」取得 MAS 標章必要安全開發實務共計 37 項，詳見表 16 所示。
- 本指引區分之 4 個實務類別/共 128 項實務準則建立快速索引詳見第 102~115 頁所示，以利使用者快速閱覽查詢及利用。

表15 行動應用 App 基本資安規範技術要求事項與本指引安全開發實務對應表

節編號	資訊安全技術要求事項(基本資安規範編號)	安全分類			安全開發實務(編碼)	合計項次
		一	二	三		
3.1.1	A.行動應用程式發布(4.1.1.1.)	V	V	V	CPA-01~CPA-05	5
3.1.2	B.行動應用程式更新(4.1.1.2.)	V	V	V	CPB-01~CPB-03	3
3.1.3	C.行動應用程式安全性問題回報(4.1.1.3.)	V	V	V	CPC-01~CPC-02	2
3.1.4	D.敏感性資料蒐集(4.1.2.1.)	V	V	V	CPD-01~CPD-06	6
3.1.5	E.敏感性資料利用(4.1.2.2.)	V	V	V	CPE-01~CPE-07	7
3.1.6 3.3	F.敏感性資料儲存(4.1.2.3.)	V	V	V	CPF-01~CPF-18 iOS-01~ iOS-02	20
3.1.7 3.3	G.敏感性資料傳輸(4.1.2.4.)		V	V	CPG-01~CPG-03 iOS-04	4

節編號	資訊安全技術要求事項(基本資安規範編號)	安全分類			安全開發實務(編碼)	合計項次
		一	二	三		
3.1.8	H.敏感性資料分享(4.1.2.5.)	V	V	V	CPH-01~CPH03	3
3.1.9	I.敏感性資料刪除(4.1.2.6.)	V	V	V	CPI-01~CPI06	6
3.1.10	J.付費資源使用(4.1.3.1.)			V	CPJ-01~CPJ-03	3
3.1.11	K.付費資源控管(4.1.3.2.)			V	CPK-01~CPK-06	6
3.1.12	L.使用者身分認證與授權(4.1.4.1.)		V	V	CPL-01~CPL-08	8
3.1.13	M.連線管理機制(4.1.4.2.)		V	V	CPM-01~CPM-08	8
3.1.14 3.2 3.3	N.防範惡意程式碼與避免資訊安全漏洞(4.1.5.1.)	V	V	V	CPN-01~CPN-07 Android-01~Android-08 iOS-03	16
3.1.15	O.行動應用程式完整性(4.1.5.2.)			V	CPO-01~CPO-05	5
3.1.16	P.函式庫引用安全(4.1.5.3.)	V	V	V	CPP-01~CPP-02	2
3.1.17 3.2	Q.使用者輸入驗證(4.1.5.4.)	V	V	V	CPQ-01~CPQ-10 Android-09~Android-10	12
3.4	作業系統強化與記錄留存	V	V	V	Server-01~Server-02	2
3.4	網頁服務安全	V	V	V	Server-03~Server-08	6
3.4	網路安全防護	V	V	V	Server-09~Server-12	4
					合計(項)	128

資料來源：經濟部工業局行動應用 App 基本資安規範及本計畫整理

表16 符合「行動應用 App 基本資安檢測基準」取得 MAS 標章必要安全開發實務

節編號	基本資安檢要求事項(基本資安規範編號)	安全分類			安全開發實務(編碼)	合計項次
		一	二	三		
3.1.1	A.行動應用程式發布(4.1.1.1.)	V	V	V	CPA-01	1
3.1.2	B.行動應用程式更新(4.1.1.2.)	V	V	V	N/A	0
3.1.3	C.行動應用程式安全性問題回報(4.1.1.3.)	V	V	V	CPC-01	1
3.1.4	D.敏感性資料蒐集(4.1.2.1.)	V	V	V	CPD-05,CPD-06	2
3.1.5	E.敏感性資料利用(4.1.2.2.)	V	V	V	N/A	-
3.1.6 3.3	F.敏感性資料儲存(4.1.2.3.)	V	V	V	CPF-01,CPF-02, CPF-05,CPF-07, CPF-09,CPF-10, CPF-12	7
3.1.7	G.敏感性資料傳輸(4.1.2.4.)		V	V	N/A	-
3.1.8	H.敏感性資料分享(4.1.2.5.)	V	V	V	CPH-01,CPH02	2
3.1.9	I.敏感性資料刪除(4.1.2.6.)	V	V	V	N/A	-
3.1.10	J.付費資源使用(4.1.3.1.)			V	CPJ-01	1
3.1.11	K.付費資源控管(4.1.3.2.)			V	CPK-01,CPK-03	2
3.1.12	L.使用者身分認證與授權(4.1.4.1.)		V	V	CPL-01	1
3.1.13	M.連線管理機制(4.1.4.2.)		V	V	CPM-01,CPM-02, CPM-03,CPM-04, CPM-05,CPM-06,	7

節編號	基本資安檢要求事項(基本資安規範編號)	安全分類			安全開發實務(編碼)	合計項次
		一	二	三		
					CPM-07	
3.1.14	N.防範惡意程式碼與避免資訊安全漏洞(4.1.5.1.)	V	V	V	CPN-01,CPN-02,CPN-03,CPN-04	4
3.1.15	O.行動應用程式完整性(4.1.5.2.)			V	N/A	-
3.1.16	P.函式庫引用安全(4.1.5.3.)	V	V	V	CPP-01	1
3.1.17 3.2	Q.使用者輸入驗證(4.1.5.4.)	V	V	V	CPQ-01,CPQ-05,CPQ-06,CPQ-07,CPQ-08,CPQ-09 Android-09,Android-10	8
3.4	作業系統強化與記錄留存	V	V	V	N/A	-
3.4	網頁服務安全	V	V	V	N/A	-
3.4	網路安全防護	V	V	V	N/A	-
					合計	37

資料來源：經濟部工業局行動應用 App 基本資安檢測基準及本計畫整理

實務準則快速索引

3.1.共通安全開發實務準則

3.1.1. 行動應用程式發布(A)

CPA-01:於行動應用程式商店提供及應用程式內實作，提供隱私權政策說明連結，說明欲存取之敏感性資料、行動裝置資源及宣告權限用途。

說明：

使用者至行動應用程式商店下載安裝行動應用 App 時，除了考量功能及價格因素外，還會考量隱私的議題。例如：使用者可能會擔心安裝的免費行動應用 App 是否會在背景中存取使用者的連絡人資料、地理位置資訊、社群的對話及使用時間等敏感資訊，並將它們傳送至使用者不知道的遠端資料庫，如果有此疑慮，使用者可能會選擇不安裝此 App，所以需事前規劃並撰寫此行動應用 App 的隱私權政策(或稱資料政策)，至少需包括但不限於：

- 蒐集資料類別有那些？如連絡人、像片、影音、地理位置、社群互動、付費、裝置資訊及第三方資訊資訊等。
- 如何利用蒐集到的資料，如加值應用、廣告行銷、客戶服務、改善安全和防護等

如何分享蒐集到的資料，分享的對象(包含個人、企業及政府或第三方等)、分享的方式，如透過 API、後端企業間資料庫對資料庫、使用者間點對點等。

而除了在隱私權說明之外，現行的 App 存取特殊資料時，不管是系統層要求或開發人員客製，也應實作存取要求、警示及選項予使用者。

隱私權政策的參考範本如下：

1.隱私權保護政策的適用範圍

請說明該 App/網站隱私權保護政策的適用範圍。

2.個人資料的蒐集、處理及利用方式

請說明該 App/網站針對個人資料的蒐集、處理及利用方式。

3.資料之保護

請說明該 App/網站針對所蒐集資料的安全維護措施，如防止個人資料被竊取、竄改、毀損、滅失或洩漏，採取技術上及組織上之必要措施。

4.網站對外的相關連結

請說明對外連結網站不適用本網站的隱私權保護政策，必須參考該連結 App/網站中的隱私權保護政策。

5.與第三人共用個人資料之政策

6.Cookie 之使用

請說明該 App/網站使用 Cookie 技術用以儲存或追蹤使用者的資料的情況。

7.隱私權保護政策之修正

請說明該 App/網站針對隱私權保護政策之修正措施。

另可參考 Center for Democracy & Technology(CDT)的行動應用程式開發最佳實務的隱私權宣告範本(https://www.cdt.org/files/pdfs/Apps_Best_Practices_v_beta.pdf （查詢時間：2016/10/1))

並應實作在應用程式內，以 iOS 版 facebook 的 App 為例：

上架至行動應用程式商店時，以 iOS 的 App Store 為例：

點選圖標下的「隱私權原則」連結，可以進一步在網頁瀏覽器檢視相關資訊

在 Google Play 商店資訊中加入隱私權政策步驟如下：

1. 前往 Google Play Developer Console（查詢時間：2016/10/1）。
2. 選取應用程式。
3. 選取[商店資訊]。
4. 在「隱私權政策」之下，輸入您的隱私權政策所在的網址。
5. 選取[儲存草稿](新應用程式)或[上傳更新](現有應用程式)。

在 App Store 加入隱私權政策步驟如下：

需要編輯 iTunes Connect App 屬性中的版本資訊，隱私權政策的 URL 為選填，本指引建議應填寫，版本資訊各欄位說明如下表：

「版本資訊」屬性包含在商店中顯示的本地化文字和影像。

對版本層級 App 資訊所做的更動會立即發布；更改的內容約需 24 小時才會完全在 Store 中更新。

如需有關設定初始值的資訊，請參閱「為 App 建立 iTunes Connect 記錄」（查詢時間：2016/10/1）。如需有關更新這些值的資訊，請參閱「檢視與更改 App 的 metadata」（查詢時間：2016/10/1）。

許多各商店地區的「版本資訊」屬性，都可在 App 的「版本摘要」中查看。請參閱「檢視各地區的版本詳細資料」（查詢時間：2016/10/1）。

屬性	描述	可編輯
語言	此 metadata 的語言。	已鎖定
螢幕快照	每個本地化版本都必須有一張螢幕快照，最多可再加入 4 張。如需螢幕快照的詳細規格，請參閱「螢幕快照屬性」(查詢時間：105/10/1)。	已鎖定
App 預覽	App 預覽為示範 App 用法的短片，可以省略。您可針對每種裝置各提供一段 App 預覽。如需 App 預覽的詳細規格，請參閱 「App 預覽屬性」(查詢時間：105/10/1)。	已鎖定
名稱	App 顯示在商店中的本地化名稱。App 名稱至少要有 2 個字元，長度不超過 75 個位元組。	已鎖定
描述 (必填)	關於 App 的本地化描述，詳細介紹特色和功能。描述長度不得超過 4000 位元組。描述必須為純文字，並加上必要的斷行標記。HTML 格式將無法辨識。請務必檢查文字內容是否有拼字或文法錯誤。	已解鎖
此版本的新增功能 (必填)	本地化的版本附註，詳細介紹此 App 版本的更動內容。例如：您可以列出新增功能、UI 改進或錯誤修正。此處文字長度上限為 4000 位元組。 第一版的 App 不會顯示此欄位。	已解鎖
關鍵字 (必填)	描述 App 的一或多個本地化關鍵字。以逗號分隔搜尋詞彙。至少需要一個大於 2 個字元的關鍵字。最多可提供 100 個位元組的內容。可透過 App 名稱或公司名稱搜尋，因此無需在關鍵字列表中重複列出這些值。不可輸入其他 App 或公司的名稱。	已鎖定
支援 URL (必填)	您計劃為對 App 有疑問之使用者提供的支援網站。支援 URL 必須指向真實的聯絡資訊，以便使用者可就 App 問題、一般意見和功能強化要求與您聯絡。URL 可指定本地化網站。 請輸入完整的 URL，包括通訊協定在內。例如： http://support.example.com(查詢日期：2016/10/1)。	已解鎖

屬性	描述	可編輯
行銷 URL (可不填)	使用者可取得更多關於 App 資訊的網站。URL 可指定本地化網站。 請輸入完整的 URL，包括通訊協定在內。	已解鎖
隱私權政策 URL (可不填)	連結至貴公司隱私權政策的 URL。所有提供可自動續訂或免費訂閱的 App，以及設定為「專為兒童製作」的 App，皆須輸入「隱私權政策 URL」。客戶在發票以及收到的訂閱確認電子郵件中會看見此 URL。URL 可指定本地化網站。 請輸入完整的 URL，包括通訊協定在內。 請注意：若 App 設定為「專為兒童製作」，則必須為 App 提供的每個本地化版本都指定一個「隱私權政策 URL」。	已解鎖

安全開發生命週期：需求階段、開發實作階段、部署維運階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.1.1. 行動應用程式發布

行動應用 App 基本資安檢測基準：

4.1.1.1.2. 檢查行動應用程式是否於可信任之應用程式商店，說明欲存取之敏感性資料、行動裝置資源及宣告權限用途。

各國開發要點參考： JSSEC AASDCG 5.5

參考來源：

CDT：

BEST PRACTICES FOR MOBILE APPLICATIONS DEVELOPERS（查詢時間：2016/10/1)

Android：

https://support.google.com/googleplay/android-developer/answer/113469（查詢時間：2016/10/1)

iOS：

https://developer.apple.com/library/ios/documentation/LanguagesUtilities/Conceptual/iTunesConnect_Guide_zh_TW/Chapters/Properties.html（查詢時間：2016/10/1)

CPA-02:行動應用 App 實際權限應與於行動平台商店提供及 App 內宣告終端使用者授權約定(EULAs)、應用程式說明、程式內部通知及與 CPA-01 於欲存取之敏感性資料、行動智慧裝置資源及宣告權限用途一致。

說明：

在 CPA-01 中在行動平台商店提供及 App 宣告終端使用者授權約定(EULAs)、應用程式說明、程式內部通知等皆是明示使用者行動應用 App 預期行為，在程式開發人員依程式設計規格書設計實際存取資料(包含敏感性資料)範圍及調用行動智慧裝置資源權限應為一致，為維持一致性在行動應用 App 安全開發生命週期中，專案經理、系統分析人員、系統設計人員、開發人員及測試/品管人員均需將此一致性要求檢查與確認納入必要工作項目，確保維持一致性。

若官網有 QRCode 下載連結或網址連結，也應於頁面旁載明標的基本資訊，包含應用程式名稱、版本及實際下載位置。

安全開發生命週期：需求、設計、開發實作及測試、部署維運階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.1.1. 行動應用程式發布

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：NIST 800 SP-163 3.1.3

參考來源：Do you need an End User Licence Agreement (EULA) for an App? (查詢時間：2016/10/1)

CPA-03:應於行動應用 App 上架前確保內部軟體品質流程及版本控制均已實作完成。

說明：

當使用者想要尋找能協助其工作、娛樂或其他應用類別的某項功能 App，會連線到行動平台商店以功能屬性或關鍵字搜尋，通常會發現有多個類似 App 可以選擇，使用者常常會參考其他下載過使用者的評論以決定要下載那一款。

如果在行動應用 App 上架前，沒有經過內部軟體品質流程確實控管品質及安全性風險，很可能會發生當使用者使用 App 開啟時發生閃退、運行緩慢或人機介面的使用者體驗(User Experience, UX)不佳，甚至因為安全性漏洞，造成使用敏感性資料外洩，使用者會在行動平台商店 App 下載頁面留下負面評論，最終讓 App 乏人問津而下架。

在中、大型企業或軟體開發商，多半已具備軟體品質管理流程及版本控制，程式開發人員通常只會進行單元測試，為加速產品生產效率，儘可能將測試自動化，需要由品管/測試人員以整合性測試發現問題，另由資安人員以工具或人工 code reivew 方式檢測是否有安全性漏洞，如有，再由程式開發人員修改，直到所有問題都被修正，完成測試後再上傳至行動平台商店審核，減少因 App 軟體品質問題及安全性漏洞而被平台退回的次數與時間。

對小型開發商及個人獨立開發人員，至少做到開發人員相互測試及 code reivew，或使用整合性開源測試平台/工具，如 MoBSF，並確實依發現問題進行反覆測試及修正。

為使 Android 版本 App 可以提高不同廠牌 Android 行動智慧裝置相容性、可靠度及安全性，開發商除了自行建置多部不同廠牌手機測試環境，亦可運用 Google Play 平台建置雲端多品牌行動智慧裝置測試服務，測試內容包括可辨識當機次數、顯示問題和安全性漏洞。測試完成可於 Google Play Developer Console 檢視「正式發佈前測試報告」。詳細步驟請參考「運用正式發佈前測試報告找出問題」(查詢時間：2016/10/1)，下圖為「正式發佈前測試報告」實例截圖。

資料來源：Google Play Developer Console (查詢時間：2016/10/1)

除了測試之外內部版本控制亦是維持行動應用 App 軟體品質持續提升的重要活動，這邊所指的是 App 開發專案內原始程式碼版本(Build)控制，與 CPA-05 的上架期間的版本(Version)控制不同，如果程式開發團隊以敏捷式開發或是 DevOps 進行持續整合及

持續發布或當多人共同維護同一個行動應用 App 專案時，需要有整合管理工具來管理個別應用程式模組及所依賴的函式庫、API 等，管理的方式與工具與 Web App 開發類似，可以共用開發流程及工具，常見的工具有 Android Studio、Xcode、SVN 及 Github 等。

作為行動應用 App 專案的專案經理及開發主管應確認以上軟體品質管理及版本控管有效被落實在行動應用 App 的生命週期。

安全開發生命週期：開發實作、測試階段、部署維運階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.1.1. 行動應用程式發布

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：CSA MAST3.3

參考來源：

Android 開發專案內版本管理請參考 Configure Your Build（查詢時間：2016/10/1)

iOS 開發專案內版本管理請參考：應用發布指南(App Distribution Guide)（查詢時間：2016/10/1)

CPA-04:確保應用程式規格遵循行動應用商店，如蘋果的 App Store 和 Google Play 規範的規格。

說明：

為維護行動應用 App 品質與符合法令規範要求並促進行動應用 App 創新及行銷推廣 Google Play 及 App Store 皆訂有相關開發人員規範，如下：

Andorod Google Play:

開發人員政策中心（(查詢時間：2016/10/1)）：

Google Play 開發人員發布協議(查詢時間：2016/10/1)

iOS App Store ：

App Store 行銷準則(查詢時間：2016/10/1)

App 開發人員指南(查詢時間：2016/10/1)

APP 發布指南(查詢時間：2016/10/1)

App Store 審核指南(查詢時間：2016/10/1)

安全開發生命週期：部署維運階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.1.1. 行動應用程式發布

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：CSA MAST 3.3

參考來源：N/A

CPA-05:執行應用程式在產品上架期間中的版本控制。

說明：

考慮使用者行動智慧裝置硬體及行動作業系統版本的 API 相容性，使用者需要參考發布行動應用 App 的釋出版本(release)決定是否要升級，故在產品上架期間的版本控制的重要性如下：

- 使用者需要對已安裝在他們的設備和可用於安裝的升級版本的應用程式版本的具體資訊。
- 其他應用程式 - 包括發布作為一個套件中的其他應用程式 - 需查詢資訊為您的應用程式的版本，以確定其相容性，並確定其相關性。
- 通過它將發布的應用程式服務可能還需要查詢您的應用程式版本，以便能夠顯示版本給使用者。發布服務可能還需要檢查應用程式的版本，以確定相容性和建立升級的關係。
- 行動應用 App 專案經理必須協同開發人員及 App 上架人員執行應用程式在產品上架期間中的版本控制。

Android 的版本控制作法請參考 Android Studio 的 Version Your App(查詢時間：2016/10/1)

iOS 的版本控制作法請參考 xcode 的 Using Source Code Control(查詢時間：2016/10/1)

安全開發生命週期：部署維運階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.1.1. 行動應用程式發布

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：CSA MAST 3.3

參考來源：

Android：Android Version Control(查詢時間：2016/10/1)

3.1.2. 行動應用程式更新(B)

CPB-01:行動應用 App 應提供功能性與安全性更新，以因應新功能加入、發現漏洞及平台安全性提升之需求。

說明：

行動應用 App 於行動平台商店上線後，使用者自平台付費或免費取得後會持續使用，在行動應用 App 存續的生命週期期間可能會因為下列因素有更新需求：

- 使用者意見回饋。
- 行動應用 App 缺陷。
- 增加新功能。
- 安全漏洞修補。
- 行動裝置作業系統更新。
- 隱私權政策更新。

專案經理及開發人員應因應上述更新需求，應提供行動應用 App 功能性與安全性更新版本，並將更新資訊以電子郵件或行動應用 App 提示，推播給使用者，提醒使用者更新。

另外需考量是接到需求、進行開發、上傳給平台審核到釋出需要時間，也意味安全性的漏洞有一定的空窗期，Andorid 目前最新的作業系統發布版本為 6.0(代號為 **Marshmallow)，2016/3/9** 日發布 7.0 版本 Google 發布 Android N 的開發人員預覽版本，近 3 個版本功能性及安全性更新摘要如下：

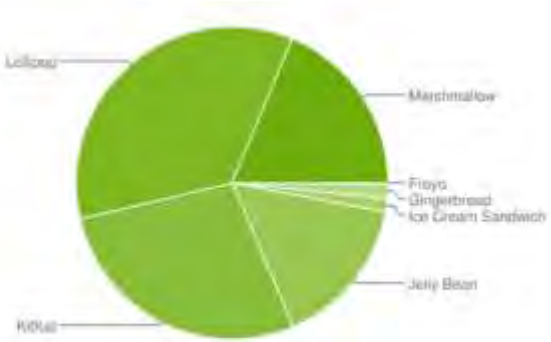
5.1/5.1.1 (Lollipop) 基於 Linux Kernel 3.4.0	2015/3/10 Google 釋出 Android 5.1(棒棒糖 Lollipop)。主要更新如下： •對多卡和高畫質語音的支援。 •快速設定 Wi-Fi 和藍芽。 •通過「裝置保護」功能帶來更好的安全性。 •最佳化音量調節。 2015/4/22 Google 釋出 Android 5.1.1(棒棒糖 Lollipop)。主要更新如下： •修復記憶體洩漏(查詢日期:2016/10/1)(memory leak)(的問題。
6.0/6.0.1 (Marshmallow) 基於 Linux Kernel 3.4.0	2015/5/29 Google 於 I/O 大會上展示 Android M 的開發人員預覽版本，並開放下載。 2015/10/5 Google 釋出 Android 6.0(棉花糖 Marshmallow)。主要更新如下： •限制 App 存取權限。 •增加 Chrome Custom Tab 功能。 •增加 Android Pay 付費功能。

	•原生系統支援指紋識別功能，用於裝置解鎖及在 Play 商店代替輸入密碼。 •新增 Doze 功能，可視環境調整裝置效能，而減少電源的消耗(Google 宣稱能比之前的版本的 Nexus 5 和 Nexus 6 省電 30%)。 •支援 USB Type C，可進行更快速的充電。 •簡化音量控制的操作。 •其他細節改善。 2015/12/7 Google 釋出 Android 6.0.1(棉花糖 Marshmallow)。主要更新如下： •加入 200 個表情符號。 •「零打擾」模式中，讓「直到下次鬧鈴響時重新啟動」的選項回歸。 •修復系統錯誤。
7.0 (Android Nougat) Developer Preview (4)	2016/3/9 Google 發布 Android N 的開發人員預覽版本，並開放下載。 主要更新如下： •支援多視窗模式。 •加入應用在 Android Wear 上的 RemoteInput notification API。 •強化 Doze 的省電功能。 •加入暗色主題。 •強化 Smart Lock 功能。 •Recent App 增加全部清除功能。 •加入新版 Emoji。 •支援 Vulkan。 •更新 App Folder。

資料來源：Wiki （查詢時間：2016/10/1)

但因為 Android 是 Google 授權給各家行動智慧裝置硬體商預載的作業系統，Android 系統可升級至何種版本，需要由各硬體廠商視機型及內部軟硬體整合，開發新的函式庫進度而定。如 Andorid 6.0 在 2015 年 10 月釋出，但在 2016 年 2 月的統計資料，只有 1.2%的 Andorid 裝置升級到 6.0，34%為 5.0 及 5.1，將近有 65%的 Android 仍在 4.4 以前的版本。

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	1.5%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.4%
4.1.x	Jelly Bean	16	5.6%
4.2.x		17	7.7%
4.3		18	2.3%
4.4	KitKat	19	27.7%
5.0	Lollipop	21	13.1%
5.1		22	21.9%
6.0	Marshmallow	23	18.7%

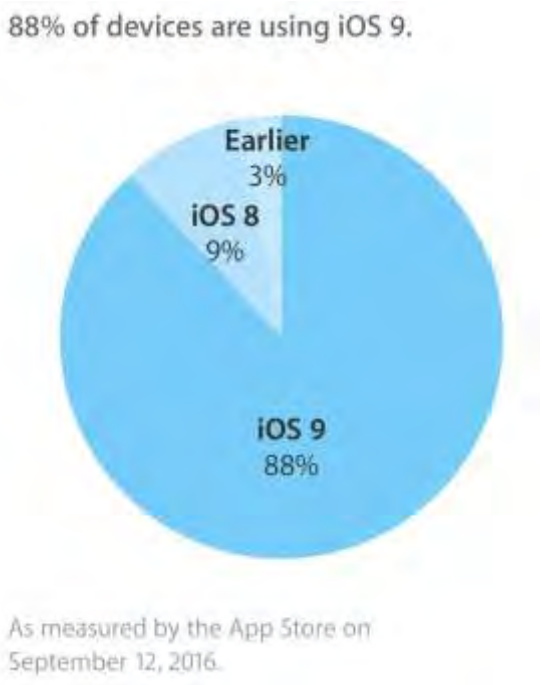


Data collected during a 7-day period ending on September 5, 2016.
Any versions with less than 0.1% distribution are not shown.

資料來源： 平台版本的儀表板（查詢時間：2016/10/1)

Android 的開放性為開發人員提供了很大的發揮空間，更多的權限也意味著容易被誤用。幾乎每一個 Android 裝置使用者都有過對應用程式自動啟動，濫用權限的困擾。Android 6.0（API Level 23）在權限管理方式、應用程式許可、記憶體管理等方面都進行很大的修改。只要開發人員按照系統規範來開發應用程式，可以大幅改善使用者的體驗及安全性。

iOS 因為是專屬硬體的專屬作業系統，Apple 並未開放給第三方硬體使用，iOS 最新為 9.X 版，依 App Store 2016/7/18 測得資料，86%iOS device 已為 iOS 9，11%為 iOS 8，只有 3%是 iOS 7 以前的版本，Apple 已於 2016 年 9 月釋出 iOS 10 的正式版本，對 iOS 開發人員而言相較 Andorid 單純許多，97%都在最新的兩個版本(版本資訊，請參考 iOS 版本歷史(查詢時間：2016/10/1))，依非 iOS 官方網站 David-smith 統計，由美國時間 2016/9/13 正式發布 iOS 10 至 2016/9/24 止已有 28.4% iOS 裝置升級至 iOS 10（請參考 DavidSmith(查詢時間：2016/10/1)）。



資料來源：App Store(查詢時間：2016/10/1)

開發人員應因應作業系統平台更新所提供功能提升及安全性改善，提供使用者更新的行動應用 App 版本。參照 CPA-01，當更新隱私權政策，也要通知使用者。

iOS 重大版本更新時，Apple 都會釋放重要資訊，包括新功能介紹、安全更新、SDK 更新說明，請參考：

WhatsNewIniOS -Introduction.html - apple developer(查詢時間：2016/10/1)

Apple security updates (查詢時間：2016/10/1)

iOSSDK-7.0 release notes iOSSDK-8.0 release notes iOSSDK-9.0 release notes (查詢時間：2016/10/1)

開發人員須注意重大更新事項，尤其與安全或隱私有關之議題，例如：iOS 7.0 之 UUID，iOS 8.0 之 WKWebView 及 iOS 9.0 之 App Transport Security 等。

安全開發生命週期：部署維運階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.1.2.行動應用程式更新

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 9.1

參考來源：N/A

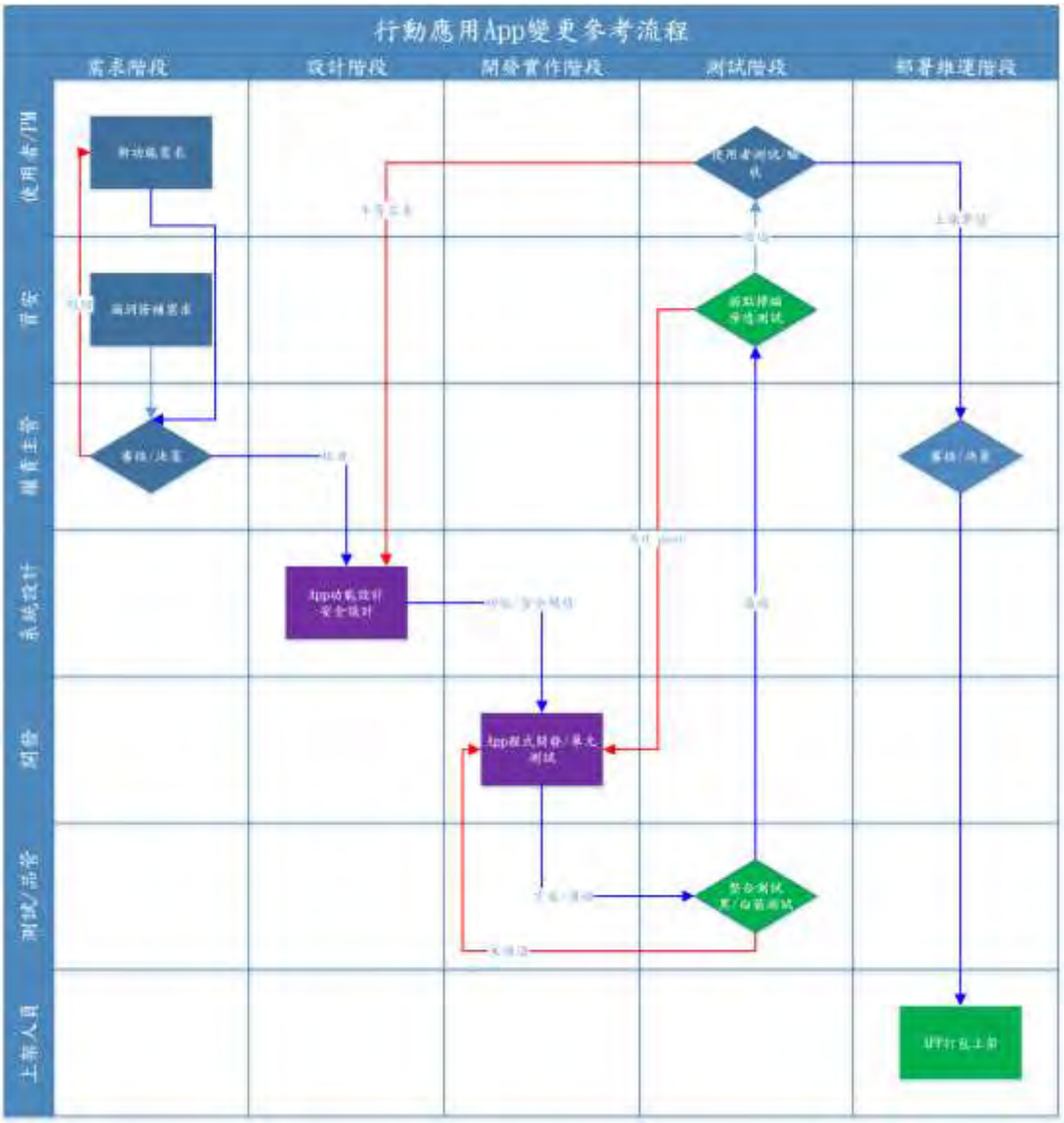
CPB-02:應建立內部行動應用 App 變更管理程序。

說明：

因應 CPB-01 行動應用 App 在平台商店更新釋出，並考量 CPA-03 軟體品質及版本控制要求，中大型開發商及小型或個人工作室，均需要建立適當的變更管理流程，以管制變更的需求可以及時被完成。

通常新的平台作業系統版本正式釋出前都會先釋出開發人員體驗版本，讓開發人員可以自行因應新版本提出新功能及安全性修改，可以先進行開發及測試，當使用者的行動裝置升級作業系統後立即有行動應用 App 更新版本可以使用。

變更管理流程視每個組織及開發單位人數及分工，沒有一定流程，只要能確認每項從變更需求至變更的結果的過程可以雙向追蹤即可，中大型組織參考流程如下圖：



資料來源：本計畫整理

上圖中如果開發團隊人數及分工不如中大型組織的小型或個人工作室可以考慮相同顏色程序負責人員可以合併兼任，如果針對安全性漏洞修補，更是透過內部變更管理程序，可以有效管制漏洞及時被修補，縮短被惡意攻擊的空窗期。

開發單位主管或是利害關係人員(客戶或稽核人員)應確認內部行動應用 App 變更管理程序是否已被建立且落實執行。

安全開發生命週期：需求、設計、開發實作、測試、部署維運階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.1.2.行動應用程式更新

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：CSA MAST 3.4

參考來源：N/A

CPB-03:所有修改版本，應檢查是否已符合行動應用平台商店，如 App Store 和 Google Play 建立的更新要求事項。

說明：

完成修改的行動應用 App，可運用程式碼比對工具，如 Android Studio 內建比對工具，搭配 Xcode 使用的 FileMerge，將修改部分標示出來以縮短檢查所需的工時及人力，針對修改版本在上線前可參考 CPA-04 規範，進行核對，由行動應用 App 品管人員或開發主管核確認後，再上傳至行動平台商店審核。

此檢查活動應設計納入 CPB-02 內部變更管理流程，以確保修改行動應用 App 符合 App Store 和 Google Play 建立的更新的要求。

安全開發生命週期：部署維運階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.1.2.行動應用程式更新

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：CSA MAST3.4

參考來源：N/A

3.1.3. 行動應用程式安全性問題回報(C)

CPC-01:於可信任之應用程式商店或行動應用程式內，提供聯絡網頁、電子郵件、電話或其他類型聯絡方式。

說明：

在 CPA-01 中的 Google Play 的商店資訊及 iTunes Connect App 屬性的版本資訊中開發人員的聯絡網頁、電子郵件、電話或其他類型聯絡方式是必填欄位，如果聯絡方式有更新，Android 可以至 Google Play Developer Console 新增或修改，步驟如下：

1. 前往 Google Play Developer Console (查詢時間：2016/10/1)。
2. 選取 [所有應用程式] 。
3. 選取應用程式。
4. 在「聯絡資訊」之下，加入您的聯絡方式。您可以提供多種支援管道(網站、電子郵件、電話)，但必須提供聯絡電子郵件才能在 Google Play 發布應用程式。

iOS 如要修改聯絡資訊，需在當 App 版本的狀態為「銷售準備就緒」時，可更改下列 App metadata：

- 描述。
- 此版本的新增功能。
- 支援 URL、行銷 URL、隱私權政策 URL。
- 許可協議。
- 版權。
- 貿易代表聯絡資訊。
- App 地區範圍檔案。

如果是中大型開發商，建議使用客服專線或是公務用的固定聯絡方式，避免以登記個別開發人員之聯絡資訊，以避免當該開發人員職位異動時，開發商無法接收到客訴或安全性通報資訊。

安全開發生命週期：部署維運階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.1.3.行動應用程式安全性問題回報

行動應用 App 基本資安檢測基準：

4.1.1.3.1.檢查行動應用程式是否於可信任之應用程式商店或行動應用程式內，提供聯絡網頁、電子郵件、電話或其他類型聯絡方式，並可實際聯絡成功。

各國開發要點參考：ENISA SSDG9.3

參考來源：N/A

CPC-02:行動應用 App 開發團隊，應關注作業系統平台或安全社群或常見弱點通報(CVE)之相關弱點通報，改善行動應用 App 安全性。

說明：

行動應用平台程式商店會持續監控 App 的安全狀況，並能夠在事件發生的情況下，可在短時間內遠端進行刪除 App。因此，通過官方 App 商店發布應用程式，可以提供 App 發現嚴重安全性漏洞時的安全網。

除了行動應用平台程式商店監控通報外，App 開發人員或資安人員也應訂閱行動應用平台官方或 CVE 漏洞修補資訊，如下

Android：Android 安全性公告(查詢時間：2016/10/1)

iOS：Apple security updates(查詢時間：2016/10/1)

另外也可以參考 Common Vulnerabilities and Exposure 資料庫，以 Android 或 iOS 為關鍵字去搜尋。

CVE：for Android(查詢時間：2016/10/1)

CVE：for iOS(查詢時間：2016/10/1)

當收到前述漏洞通報，應由資安人員或是系統設計人員應進行確認及風險評估，已上架及開發中行動應用 App 是否有修補必要，如是應納入 CPB-02 內部變更流程。

安全開發生命週期：部署維運階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.1.3.行動應用程式安全性問題回報

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 9.2

參考來源：

Android：Android 安全性公告(查詢時間：2016/10/1)

iOS：Apple security updates(查詢時間：2016/10/1)

3.1.4. 敏感性資料蒐集(D)

CPD-01:需求階段需要識別計畫開發行動應用 App 蒐集、處理及利用敏應資料類別(例如：密碼，個人資料、地理位置、財務資訊及錯誤日誌等)及行動應用 App 平台與發行國的隱私法令要求，定義安全性需求。

說明：

通常免費或無償提供行動應用 App，多數負有蒐集、利用及分享行動智慧裝置擁有者個人資料有關識別與行為資料任務，對於個人資料定義，查我國個人資料保護法第二條：

本法用詞，定義如下：

一、個人資料：指自然人之姓名、出生年月日、國民身分證統一編號、護照號碼、特徵、指紋、婚姻、家庭、教育、職業、病歷、醫療、基因、性生活、健康檢查、犯罪前科、聯絡方式、財務情況、社會活動及其他得以直接或間接方式識別該個人之資料。

二、個人資料檔案：指依系統建立而得以自動化機器或其他非自動化方式檢索、整理之個人資料之集合。

三、蒐集：指以任何方式取得個人資料。

四、處理：指為建立或利用個人資料檔案所為資料之記錄、輸入、儲存、編輯、更正、複製、檢索、刪除、輸出、連結或內部傳送。

五、利用：指將蒐集之個人資料為處理以外之使用。

六、國際傳輸：指將個人資料作跨國(境)之處理或利用。

七、公務機關：指依法行使公權力之中央或地方機關或行政法人。

八、非公務機關：指前款以外之自然人、法人或其他團體。

九、當事人：指個人資料之本人。

以上第一款說明了個人資料類別，第二～六款也說明個人資料檔案型態蒐集、處理及利用及國際傳輸的定義。

第 6 條：有關病歷、醫療、基因、性生活、健康檢查及犯罪前科之個人資料，不得蒐集、處理或利用。

第 6 條說明了病歷、醫療、基因、性生活、健康檢查及犯罪前科之個人資料，原則上禁止蒐集，除了符合第 6 條中的但書，如法律規定、去識別化學術研究及當事人明確表達同意等例外情況下一律禁止蒐集。依 104/12/30 修訂最新個人資料保護法，例外情形如下：

一、法律明文規定。

二、公務機關執行法定職務或非公務機關履行法定義務必要範圍內，且事前或事後有適當安全維護措施。

三、當事人自行公開或其他已合法公開之個人資料。

四、公務機關或學術研究機構基於醫療、衛生或犯罪預防之目的，為統計

或學術研究而有必要，且資料經過提供者處理後或經蒐集者依其揭露方式無從識別特定之當事人。
五、為協助公務機關執行法定職務或非公務機關履行法定義務必要範圍內，且事前或事後有適當安全維護措施。
六、經當事人書面同意。但逾越特定目的之必要範圍或其他法律另有限制不得僅依當事人書面同意蒐集、處理或利用，或其同意違反其意願者，不在此限。
依前項規定蒐集、處理或利用個人資料，準用第八條、第九條規定；其中前項第六款之書面同意，準用第七條第一項、第二項及第四項規定，並以書面為之。

除了個人資料保護法，仍有像稅法、銀行法、醫師法等特別法對個人資料及其他敏感性資料規範。

在我國個人資料保護法第 8 條規定：

公務機關或非公務機關依第十五條或第十九條規導向當事人蒐集個人資料時，應明確告知當事人下列事項：
一、公務機關或非公務機關名稱。
二、蒐集之目的。
三、個人資料之類別。
四、個人資料利用之期間、地區、對象及方式。
五、當事人依第三條規定得行使之權利及方式。
六、當事人得自由選擇提供個人資料時，不提供將對其權益之影響。
有下列情形之一者，得免為前項之告知：
一、依法律規定得免告知。
二、個人資料之蒐集係公務機關執行法定職務或非公務機關履行法定義務所必要。
三、告知將妨害公務機關執行法定職務。
四、告知將妨害公共利益。
五、當事人明知應告知之內容。
六、個人資料之蒐集非基於營利之目的，且對當事人顯無不利之影響。

就是明定要直接蒐集當事人的個人資料，應明確告知由那一個公務非公務機關將會以何種目的蒐集個資當事人的何種類別資料，且會利用之期間、地區、對象及方式，及當事人對已被蒐集的個人資料有何權利，讓使用者在下載之前或是蒐集使用者本人個人資料前有判斷的依據，讓使用者可以選擇同意或是拒絕，如果選擇拒絕是否會影響公務或非公務機關對當事人服務的品質。

實務上很多免費的行動應用 App，其蒐集的使用者的敏感性資料很多都與 App 提供給使用者服務無關，屬於個人資料特定目的外之蒐集、處理及利用，需符合個人資料保護法第 16 條(公務機關適用)及第 20 條(非公務機關適用)的要件之一，包括個資當事人同意。

當 App 需要上架至各國的行動應用平台商店，應需先了解當地政府所頒發的個人資料保護法令，避免誤觸他國法網。在 Android 官方網頁有 Designed for Families 的應用程式(包括應用程式為提供服務而使用的所有 API)符合《兒童線上隱私保護法》(COPPA(查詢時間：2016/10/1)) 。

iOS 的開發人員指南分別有整理出美國(聯邦、加州)、歐盟、日本的隱私規範

Android：符合 COPPA 法規(查詢時間：2016/10/1)

iOS：Supporting User Privacy(查詢時間：2016/10/1)

行動應用 App 專案經理與系統分析人員應充分識別需求單位處理個人資料在內的敏感資料類別，並定義出安全保護需求，由系統分析/設計人員可參考隱私保護最佳實務「Privacy by Design(查詢時間：2016/10/1)」(從設計著手保護隱私)進一步設計安控及隱私保護措施。Privacy by Design 的 7 大原則如下：

一、應化被動為主動，並防患於未然，而非事後亡羊補牢（Proactive not Reactive; Preventative not Remedial）

二、使隱私成為預設機制（Privacy as the Default Setting）

三、於設計中植入隱私（Privacy Embedded into Design)

四、完整的機能：正和而非零和（Full Functionality-Positive-Sum, not Zero-Sum）

五、環環相扣的安全：貫穿資料生命週期的保護（End-to-End Security-Full Lifecycle Protection）

六、能見度與透明度：保持開放性（Visibility and Transparency-Keep it Open）

七、尊重使用者隱私：確保以使用者為中心（Respect for User Privacy-Keep it User-Centric）

在設計與開發實務有多種選擇，本指引強烈建議依上述 7 大原則作為優先考量，可完整貫穿敏感性資料生命週期、不論是在行動應用 App 本地端、後端伺服器、資料庫及網路傳送，隨時隨地落實 Privacy by Design，方能尊重使用者隱私，完善敏感性資料保護。

安全開發生命週期：需求階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.1.敏感性資料蒐集

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：JSSEC AASDCG 5.5

參考來源：Privacy by Design(查詢時間：2016/10/1)

CPD-02:建立預計提供給使用者的敏感資料蒐集範圍、類別、保存期限、使用地點及生命週期處理政策。

說明：

就個資當事人的權利如個人資料保護法第 3 條：

當事人就其個人資料依本法規定行使之下列權利，不得預先拋棄或以特約限制之：

一、查詢或請求閱覽。

二、請求製給複製本。

三、請求補充或更正。

四、請求停止蒐集、處理或利用。

五、請求刪除。

第 5 條：

個人資料之蒐集、處理或利用，應尊重當事人之權益，依誠實及信用方法為之，不得逾越特定目的之必要範圍，並應與蒐集之目的具有正當合理之關聯。

第 5 條說明個人資料運用需尊重當事人權益及意願，需以誠信及合理正當方式處理有關聯的個人資料

由於我國個人資料保護法立法時參考了經濟合作與發展組織(Organisation for Economic Co-operation and Development, OECD)的隱私權保護原則設計及「Privacy by Design(查詢時間：2016/10/1)」，所以尊重個資當事人(即行動智慧裝置與 App 使用者)意願，且需符合特定目的不逾合理、正當及有關聯範圍，對其個人資料被蒐集、利用、儲存、傳輸、分享及刪除等完整個人資料生命週期，皆需依此原則，設計在行動應用 App 中在第一次或每次需要蒐集、處理及利用時讓使用者可以充分表達同意或拒絕當事人個人資料在內的敏感性資料。

安全開發生命週期：需求階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.1.敏感性資料蒐集

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 7.1, JSSEC AASDCG 5.5

參考來源：

個人資料保護法第 3 條、第 5 條(查詢時間：2016/10/1)

CPD-03:在設計階段依行動應用 App 需求識別時，需蒐集、處理、儲存、利用、傳送及刪除敏感性資料類別與內、外部媒體儲存需求，以使用者同意及最低揭露原則設計權限安全性控制措施，如身分驗證、API 呼叫、加密儲存及傳送等安全性。

說明：

依 CPD-01 及 CPD-02 的敏感性資料蒐集、利用、儲存、傳輸、分享及刪除需求，在設計階段，根據敏感性資料分類儲存，並相應地選用於控制(例如：密碼，個人資料，位置，錯誤日誌等)。過程中，根據其分類儲存和使用的資料。驗證應用到敏感資料的 API 呼叫的安全性。

在設計時需考慮行動應用 App 的使用情境，如在公眾環境使用不安全的無線存取網路(Wi-Fi)、使用行動裝置運用創新但不夠安全的金融科技進行財務上交易等，在程式設計時為考慮存取便利，可能會在行動智慧裝置的內、外部儲存媒體臨時儲存，建立暫存檔、備份及刪除，包括個人資料在內的敏感性資料，及可能會引用第三方函式庫、API 所涉及安全風險及對應的控制措施。

在設計整個敏感性資料在行動應用 App 傳至後端伺服器的權限，以最小權限原則，只蒐集和揭露必要的資料所需的最低權限。確定在設計階段所需要的資料，它的敏感度，以及它是的集、利用、儲存、傳輸、分享的每個資料類型否是適當。

安全開發生命週期：設計階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.1.敏感性資料蒐集

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG1.1, ENISA SSDG1.10 ,ENISA SSDG1.11

參考來源：N/A

CPD-04:檢查在同一資料堆疊(Stack)來自多個蒐集敏感資料不同蒐集來源是否重疊並衝突，如有需解決衝突。

說明：

現代愈來愈多的過去單純僅有網頁服務的應用系統，因應滑世代來臨，都已經 App 化，多數知名的網路服務，都有網頁版及 App 版(Android、iOS)，都同時提供讓使用者可以在不同平台且跨裝置、隨時隨地延續相同使用者體驗。

通常這些應用服務在後端都是整合的資料平台，甚至是同一個資料堆疊(stack)，只有前端依不同裝置及使用者環境，可能會有網頁、原生 App 及 Webkit 等不同終端介面。

除非是近期推出服務，會同時考量制定統一的使用者隱私權政策，有時前述各種不同介面形式，在比較中大型的組織或是開發商，不一定都是同屬一個專案或部門，很可能由不同專案經理、系統分析、系統設計及程式開發人員負責或是在不同時期建置，沒有同步隱私權政策部分，造成相矛盾。很可能因此造成對使用者敏感性資料重複蒐集或是在資料類別不一致。

專案經理、系統分析人員應會同應用服務規劃單位，找出來自不同蒐集來源(如原生 App+WebKit 的 HTML)間有衝突的資料蒐集範圍及內存取權限不一致部分，逐一調整為一致。

安全開發生命週期：設計階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.1.敏感性資料蒐集

行動應用 App 基本資安檢測基準：N/A

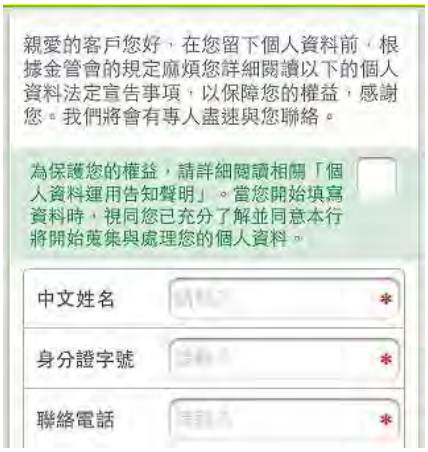
各國開發要點參考：ENISA SSDG 7.6

參考來源：N/A

CPD-05:蒐集敏感性資料應實作使用者同意或拒絕選項，提示使用者選擇時機可於(1)安裝時(2)當敏感資料被儲存或傳送前(3)預設設定為關閉同意，需使用者自行開啟。

說明：

當使用者下載並安裝行動應用 App 後，行動應用 App 不論是需要使用者註冊帳號、填寫真實姓名、聯絡方式或是讀取聯絡人資料、相片、地理位置都是蒐集個人資料行為。蒐集敏感資料行為是將使用者將個人資料帶入行動應用 App 資料管理生命週期的起始點，如何實現前述 CPD-01 及 CPD-02 都需要當事人依當事意願來決定是否進行實質的蒐集行為。最常運用的實務實作彈跳(Pop up)視窗，如下圖，或需勾選同意選項，才能繼續。



同意或拒絕的選項可於下列時機實作：

(1)安裝時

(2)當敏感資料被儲存或傳送前

(3)預設設定為關閉同意，需使用者自行開啟

以上的如搭配身分認證機制驗證後，都可視為使用者個人意願已充分表達。

在使用者選拒絕行動應用 App 蒐集的情況下，開發人員不能實作任何未獲使用者明確表達同意被蒐集的敏感性資料。

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.1.敏感性資料蒐集

行動應用 App 基本資安檢測基準：

4.1.2.1.1. 檢查行動應用程式所有蒐集之敏感性資料，是否皆已於行動應用程式內聲明並取得使用者同意。

4.1.2.1.2.(1) 檢查行動應用程式是否提供使用者拒絕蒐集敏感性資料之選項。

各國開發要點參考：ENISA SSDG 7.2, JSSEC AASDCG 4.10

參考來源：N/A

CPD-06:應依 CPD-03 實作行動應用 App 存取敏感性資料權限，不要授與過度蒐集不必要敏感性資料權限或於未告知使用者取得同意前於行動應用 App 背景運作蒐集敏感性資料活動。

說明：

開發人員在實作權限時，務必將「取得使用者同意」與「最小權限原則」兩大原則銘記在心，不要授與過度蒐集非必要敏感性資料權限或於未告知使用者取得同意前於行動應用 App 背景運作蒐集敏感性資料活動，如免費的輸入法程式，在安裝時跳出視卻需要存取如下圖中使用者的聯絡人、位置、相片/多媒體/檔案、麥克風、Wi-Fi 連線資訊、藍芽連線資訊、裝置 ID 和通話資訊，且僅有接受選項，幾乎和輸入法功能無關。

我們可以理解行動應用商店上數以萬計的免費軟體，不外希望能蒐集個人資料在內的敏感性資料，作為行銷廣告之用，其背後仍是商業營利的考量，以此設計無異讓使用者放棄其對個人資料主導權利，部分使用者為能使用行動應用 App 功能，接受其個人資料及行動裝置資源被包裹式授權給開發商或是蒐集敏感性機構。

為符合本項實務，開發人員應設計拒絕選項，可以導引至付費使用或是具體依我國個人資料保護法第 8 條(CPD-01)告知使用者特定目的，如蒐集行為在特定目的之外蒐集、處理及利用，需符合個人資料保護法第 16 條(公務機關)或第 20 條(非公務機關)各項要件，包括當事人同意為其中之一的要件。

Android 在每個應用程式的根目錄都必須包含 AndroidManifest.xml 檔案。宣示說明檔案可向 Android 系統顯示應用程式的基本資訊，也就是系統在執行該應用程式的任何程式碼之前必須具備的資訊。宣示說明可執行下列動作：

1. 為應用程式的 Java 封裝命名。封裝名稱可當成應用程式的唯一識別碼使用。
2. 描述應用程式的元件–組成應用程式的活動(Activity)、服務(Services)、廣播接收器(Broadcast receviers)和內容供應程式(Conrent Providers)。為實作每個元件的類別命名以及發布類別(Classes)的功能(例如：Classes 可處理的 Intent(查詢日期：2016/10/1)訊息)。這些宣告可讓 Android 系統了解元件為何以及可在哪些情況下啟動。
3. 決定代管應用程式元件的程序。
4. 宣告應用程式必須擁有哪些權限，才能存取 API 受保護的部分以及與其他應用程式互動。
5. 宣示說明亦可宣告其他項目必須擁有哪些權限，才能與應用程式的元件互動。
6. 列出可在應用程式執行時提供分析和其他資訊的 Instrumentation(查詢日期：2016/10/1)類別。 只有在應用程式開發及測試完成的情況下，宣示說明中才會顯示這些宣告；這些宣告會在應用程式發布之前移除。
7. 宣告應用程式要求的最低 Android API 等級。
8. 列出應用程式必須連結的函式庫。

以上說明了一個行動應用 App 會有那些元件、最低 API 等級要求、必要連結函式庫，可以於此概略了解權限設定狀態，詳細資訊可以參考應用程式宣示說明(查詢時間：2016/10/1)。

Android 6.0(API Level 23)開始，應用程式的權限有新的機制，在 API 23 之前(Android 4.x、5.x)應用程式在安裝時會要求所有會使用到的權限，如 INTERNET 網路連線、READ_CONTACT 讀取聯絡人資料或是 WRITE_EXTERNAL_STORAGE 寫入外部儲存裝置(如 SD Card)等，在 API 23 開始，權限分為兩大類型，分別為一般權限(Normal Permission)與危險權限(Dangerous Permission)。

這兩種權限同樣都要在 AndroidManifest.xml 中使用<uses-permission>宣告，不同的是，危險權限除此之外，在執行應用程式時，如果程式碼中存取了危險權限，還會出現請求權限的對話框，要求使用者允許存取資料，下圖是存取危險權限之一的拍照權限出現的對話框：

危險權限依照功能分為以下幾個組別：

序號	群組	項目	說明
1	CALENDAR 日曆	READ_CALENDAR	讀取日曆
		WRITE_CALENDAR	寫入日曆
2	CAMERA	CAMERA	相機拍照功能
3	CONTACTS 聯絡人	READ_CONTACTS	讀取聯絡人
		WRITE_CONTACTS	寫入聯絡人
		GET_ACCOUNTS	取得手機帳號
4	LOCATION 位置	ACCESS_FINE_LOCATION	取得精確位置
		ACCESS_COARSE_LOCATION	取得大約位置
5	MICROPHONE 麥克風	RECORD_AUDIO	錄製聲音
6	PHONE 電話	READ_PHONE_STATE	讀取通話狀態
		CALL_PHONE	撥出電話
		READ_CALL_LOG	讀取通話記錄
		WRITE_CALL_LOG	寫入通話記錄
		ADD_VOICEMAIL	新增語音留言
		USE_SIP	使用 SIP 網路電話
		PROCESS_OUTGOING_CALLS	存取撥出電話
7	SENSORS 感應器	BODY_SENSORS	讀取體感資料
8	SMS 簡訊	SEND_SMS	傳送簡訊
		RECEIVE_SMS	接收簡訊
		READ_SMS	讀取簡訊
		RECEIVE_WAP_PUSH	接收 WAP 推播訊息
		RECEIVE_MMS	接收多媒體簡訊
9	STORAGE 儲存	READ_EXTERNAL_STORAGE	讀取外部儲存
		WRITE_EXTERNAL_STORAGE	寫入外部儲存

危險權限若在執行過程獲得使用者授權群組其中一種權限後，便自動取得該群組內的所有權限，例如：使用者允許第 3 組中的 READ_CONTACTS 後，應用程式也一併取得 CONTACTS 群組內的 WRITE_CONTACTS 權限。

安全開發生命週期：設計、開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.1.敏感性資料蒐集

行動應用 App 基本資安檢測基準：

4.1.2.1.2.(2) 檢查在使用者拒絕敏感性資料蒐集的情況下，行動應用程式是否未檢出蒐集敏感性資料。

各國開發要點參考：CSA MAST 4.2,ENISA SSDG 7.3, YD/T 2407-2013 5.5.4.1

參考來源：

Android：

應用程式宣示說明(查詢時間：2016/10/1)

Android 6.0 的 Permission 權限設計,Hank Tom,2016/5/15(查詢時間：2016/10/1)

3.1.5. 敏感性資料利用(E)

CPE-01:不要使用設備或與其他 App 共享持久性敏感資料作為使用者識別碼，如設備 ID，最好採用隨機產生識別碼(參考 CPM-02)，並採用相同的資料最小化權限原則應用在行動應用 App 的 Session ID 及 HTTP Session ID/cookies 等。

說明：

為保持使用者良好體驗，不需要使用者持續輸入身分認證，通常以儲存 Session ID/Cookies 在使用者端用來讓 Server 端識別使用者身分用，攻擊者為了取得使用者的敏感性資料，如個人隱私資料或曾經網購物品等，獲取使用者 Session ID/Cookies 就可以使用者身分進入伺服器竊取。攻擊者常用猜測**(Session Prediction)**、連線劫持**(Session Hijacking)**及詐取固定**(Session Fixation)**。

為了避免 Session ID 因太過簡單、或容易被猜測、或是攻擊者設置偽造釣魚網站獲取設計為持久性且固定的 Seesion ID。系統設計及開發人員在 Server 端實作網頁應用程式時建議不要以持久性且固定的敏感資料，如設備 ID、身分證 ID、電話號碼等，在每次登入時更換隨機產生或是可變量的 Session ID(參考 CPM-02)，並設定逾時就清除 session ID。

在行動應用 App 的本地端的 Session ID 也要限制被其他App 存取，以最小化權限原則設計實作，傳送至伺服器端應使用 TLS 連線加密，或將 Session ID 加密傳送，至 server 端再解密，不要將 Session ID 使用 URL(GET)以明文方式來傳遞。

安全開發生命週期：設計階段、開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.2.敏感性資料利用

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 1.12

參考來源：

HTTP Session 攻擊與防護,DevCore,翁浩正(查詢時間：2016/10/1)

CPE-02:追蹤個人使用者應以使用者帳號/密碼作為識別使用者的識別碼，避免使用電話號碼作為識別碼，並實作非基於簡訊驗證的雙因素驗證或增加額外使用者確認機制。

說明：

如同 CPE-01，設計 Session ID 時如以電話號碼作為識別碼，易造成敏感性個人資料外洩。雖然行動應用 App 在點對點實作良好的連線加密，攻擊者不易進行破解，但以近期強調通訊保密功能極佳的 WhatsApp 及 Telegram 兩款點對點加密 App，被資安研究人員破解，在展示的 Yoube 影片 Telegram SS7 Attack(查詢時間：2016/10/1)，其實被攻擊的並非 Telegram 本身的漏洞，而是利用其使用電話號碼作為識別個人的機制，以攻擊全球電信服務商廣泛採用的 SS7(Signaling System Number 7)有發信系統的漏洞，讓攻擊者能夠竊聽受害者的語音通話、竊取簡訊，然而這個漏洞最可怕的地方在於，攻擊者需要的資料僅有受害者的電話號碼以及些許設備資料。

在攻擊者可以竊取受害者簡訊的情況下，基本上這個攻擊手法，可以應用在任何透過簡訊傳送重設密碼資訊的應用程式與 App 服務，竊取帳號控制權。例如：攻擊者可以在竊取 Facebook 帳號的過程中，使用 Facebook 提供的忘記密碼功能。當系統詢問受害者要以 E-Mail 或是手機簡訊傳送一次性動態認證密碼(OTP 密碼)時，選擇以手機簡訊傳送，並竊取簡訊，就能取得認證密碼，進而重新設定受害者 Facebook 帳號的密碼，並取得帳號控制權。

系統設計與開發人員設計實作時，應避免將帳號與手機號碼連結，並實作非經過簡訊傳送密碼的雙因素驗證(Two-Factor Authentication)功能，或增加額外通知使用者察覺機制如設備指定、推播確認、郵件回覆及手機以外設備確認等，以免將使用者敏感性資料曝露在風險之下。

安全開發生命週期：設計階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.2.敏感性資料利用

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：NIST 800 SP-163 3.1.1

參考來源：

只要知道你的手機號碼，駭客就可以利用 SS7 漏洞竊取你的 Facebook 帳號，李文恩、2016/6/17(查詢時間：2016/10/1)

CPE-03:不得實作未經使用者同意，使行動應用 App 可擅自修改使用者資料的行為，包括在使用者無確認情況下刪除或修改使用者連絡人資料、通話記錄、簡訊資料和多媒體簡訊資料的行為。

說明：

使用者同意行動應用 App 開發人員蒐集敏感性資料，不代表使用者已經授權可以任意處理這些敏感性資料，破壞資料完整性與正確性，如確有需要修改使用者資料，包括刪除或修改使用者連絡人資料、通話記錄、簡訊資料和多媒體簡訊資料的行為，請參考 CPD-05 及 CPD-06 實務取得使用者同意的意向再啟動。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

Android：Java

關於個資的部分(連絡人資料、通話記錄、簡訊資料)，開發人員需要檢查是否具備此授權，如果沒有，需要主動向使用者要求允許權限。

```
import android.support.v4.app.ActivityCompat;
import android.Manifest;
import android.content.pm.PackageManager;
import static android.Manifest.permission.*;
public class MainActivity extends AppCompatActivity{
  private static final int MY_REQUEST_ACCESS_CONTACTS = 5;


  @Override
  protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      int permission = ActivityCompat.checkSelfPermission(this,
              Manifest.permission.READ_CONTACTS);
      if (permission != PackageManager.PERMISSION_GRANTED) {
          //未取得權限，向使用者要求允許權限
          ActivityCompat.requestPermissions( this,
                  new String[]{READ_CONTACTS, WRITE_CONTACTS},
                  MY_REQUEST_ACCESS_CONTACTS);
      }else{
         //已有權限，可進行檔案存取
          readContacts();
      }
  }
```

```
 @Override
public void onRequestPermissionsResult(int requestCode,
        String permissions[], int[] grantResults) {
    switch (requestCode) {
        case MY_REQUEST_ACCESS_CONTACTS: {
            // 如果請求被取消的話，傳回的陣列的大小是
            if (grantResults.length > 0
&& grantResults[0] == PackageManager.PERMISSION_GRANTED) {

                // 請求被允許，接下來就可以做一些和連絡人相關的動作
                readContacts();

            } else {
                      //使用者拒絕權限，顯示對話框告知
                     new AlertDialog.Builder(this)
                           .setMessage("必須允許聯絡人權限來使用此功能")
                           .setPositiveButton("OK", null)
                           .show();

            }
            return;
        }

        // 如果有其他的權限，那就需要處理個別的 CASE
    }
  }
}
```

iOS：Objective-C

iOS 不允許 App 存取通話紀錄、簡訊資料與多媒體簡訊，但允許經使用者授權後，取得手機聯絡人相關資料，至於修改、刪除聯絡人資料，也建議須使用者確認。

針對[檢查授權、新增、修改、刪除聯絡人資料]提供相關程式碼如下。

檢查授權

```
// Check the authorization status of our application for Address Book
-(void)checkAddressBookAccess
{
    switch (ABAddressBookGetAuthorizationStatus())
    {
```

```
        // Update our UI if the user has granted access to their Contacts
        case  kABAuthorizationStatusAuthorized:
            [self accessGrantedForAddressBook];
            break;
            // Prompt the user for access to Contacts if there is no
definitive answer
        case  kABAuthorizationStatusNotDetermined :
            [self requestAddressBookAccess];
            break;
            // Display a message if the user has denied or restricted access
to Contacts
        case  kABAuthorizationStatusDenied:
        case  kABAuthorizationStatusRestricted:
        {
            UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Privacy
Warning"
                                                            message:@"Permission
was not granted for Contacts."
                                                           delegate:nil
                                                  cancelButtonTitle:@"OK"
                                                  otherButtonTitles:nil];
            [alert show];
        }
            break;
        default:
            break;
    }
}
```

新增聯絡人

```
    ABNewPersonViewController *picker = [[ABNewPersonViewController alloc]
init];
    picker.newPersonViewDelegate = self;
    UINavigationController *navigation = [[UINavigationController alloc]
initWithRootViewController:picker];
   [self presentViewController:navigation animated:YES completion:nil];
```

修改(顯示+修改) 聯絡人

```
    // Search for the person named "Appleseed" in the address book
```

```
    NSArray *people = (NSArray
*)CFBridgingRelease(ABAddressBookCopyPeopleWithName(self.addressBook,
CFSTR("Appleseed")));
    // Display "Appleseed" information if found in the address book
    if ((people != nil) && [people count])
    {
    ABRecordRef person = (__bridge ABRecordRef)[people objectAtIndex:0];
    ABPersonViewController *picker = [[ABPersonViewController alloc] init];
    picker.personViewDelegate = self;
    picker.displayedPerson = person;
    // Allow users to edit the person’s information
    picker.allowsEditing = YES;
    [self.navigationController pushViewController:picker animated:YES];
    }
```

刪除聯絡人

```
       CFErrorRef *error = NULL;
       ABAddressBookRemoveRecord(self.addressBook, (ABRecordRef)person,
error);
       if(error != NULL)
       {
           //if any error happen
           UIAlertView    *alert =[[UIAlertView alloc] initWithTitle:@"Error"
message:@"Deleting Contact" delegate:self cancelButtonTitle:@"Cancel"
otherButtonTitles:@"OK",nil];
           [alert show];
       }
       ABAddressBookSave(self.addressBook, NULL);
```

行動應用 App 基本資安規範：4.1.2.2. 敏感性資料利用

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：YD/T 2407-2013 5.5.4.2, NIST 800 SP-163 3.1.1

參考來源：

iOS：

Human Interface Guildlines,Request Permissions(查詢時間：2016/10/1)

QuickContacts demonstrates Address Book UI from(查詢時間：2016/10/1)

developer.apple.com(查詢時間：2016/10/1)

CPE-04:謹慎利用行動應用 App 的敏感性資料，不被非授權存取或傳送到未知遠端目的地。

說明：

「開發人員可以有意或無意將行動應用 App 產生敏感性資料或相關資訊(例如：NFC、GPS 位置及稽核日誌)」，轉移到目的地不當或其他應用程式，因為他們可能沒有在應用程式開發過程中考慮這些資訊的安全性或是蓄意不尊重使用者隱私資料，進行濫用。

地理位置資料(GPS 位置)，都會被記錄在主機的歷史資料中，無法被主動刪除。或會被存放在照片資料中，所以應該善意告知使用者此項資訊。

如果嚴格落實開發控制或審查機制，或是在 App 設計、開發實作階段就遵守由使用者當事人意願處理敏感性資料的良好政策及實務，就應不存在行動應用 App 使用過程中發生未經使用者同意私自傳送或導致暴露可能的敏感資料洩漏問題。

所以應該提示使用者 App 是否刪除照片之地理位置、NFC MAC 位址等及稽核日誌等資料。

請分別參考 Android 及 iOS 的下列的安全程式碼範例。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

Android：Java

在 Android 當中，圖片定位資料可自行於系統上管理，將其設定為關閉，以避免圖片定位資料被紀錄。不過一般使用者並不易明白或容易忽視有些設定。

茲提供移除圖片地理位置資料的程式碼

```
@Override
public void surfaceCreated(SurfaceHolder holder) {
    mCamera = Camera.open();

    Camera.Parameters p = mCamera.getParameters();
    //移除相機內 GPS 相關資訊
    p.removeGpsData();

    mCamera.setParameters(p);
}
```

iOS：Objective-C

在 iOS 當中，定位資料可自行於系統上管理。但相機的地理位置資料通常被使用者忽視。提供圖片地理位置資料的編輯程式碼與移除程式碼。

圖片地理位置資料編輯程式碼

```
- (void) saveImage:(UIImage *)imageToSave withInfo:(NSDictionary *)info
{
    // Get the image metadata (EXIF & TIFF)
    NSMutableDictionary * imageMetadata = [[info
objectForKey:UIImagePickerControllerMediaMetadata] mutableCopy];
    if (!imageMetadata)
    {
        imageMetadata = [[NSMutableDictionary alloc] init];
    }
    // add (fake) GPS data
    CLLocationCoordinate2D coordSF = CLLocationCoordinate2DMake(37.732711,-
122.45224);
    // arbitrary altitude and accuracy
    double altitudeSF = 15.0;
    double accuracyHorizontal = 1.0;
    double accuracyVertical = 1.0;
    NSDate * nowDate = [NSDate date];
    // create CLLocation for image
    CLLocation * loc = [[CLLocation alloc] initWithCoordinate:coordSF
altitude:altitudeSF horizontalAccuracy:accuracyHorizontal
verticalAccuracy:accuracyVertical timestamp:nowDate];
// this is in case we try to acquire actual location instead of faking it with the
code right above
    if ( loc ) {
        [imageMetadata setObject:[self gpsDictionaryForLocation:loc]
forKey:(NSString*)kCGImagePropertyGPSDictionary];
    }

    // Get the assets library
    ALAssetsLibrary *library = [[ALAssetsLibrary alloc] init];

    // create a completion block for when we process the image
    ALAssetsLibraryWriteImageCompletionBlock imageWriteCompletionBlock =
    ^(NSURL *newURL, NSError *error) {
        if (error) {
            NSLog( @"Error writing image with metadata to Photo Library: %@", error );
        } else {
```

```
            NSLog( @"Wrote image %@ with metadata %@ to Photo
Library",newURL,imageMetadata);
        }
    };

    // Save the new image to the Camera Roll, using the completion block defined just
above
    [library writeImageToSavedPhotosAlbum:[imageToSave CGImage]
                                 metadata:imageMetadata
                          completionBlock:imageWriteCompletionBlock];
}
```

圖片地理位置資料清除

```
-(void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info
{
    NSString *mediaType = [info
                           objectForKey:UIImagePickerControllerMediaType];
    [self dismissModalViewControllerAnimated:YES];
    if ([mediaType isEqualToString:(NSString *)kUTTypeImage]) {
        UIImage *image = [info
                          objectForKey:UIImagePickerControllerOriginalImage];
        //got image, put image to imageView
        imageView.image = image;
        if (newMedia)
            UIImageWriteToSavedPhotosAlbum(image,
                                           self,
@selector(image:finishedSavingWithError:contextInfo:),
                                           nil);
    }
    else if ([mediaType isEqualToString:(NSString *)kUTTypeMovie])
    {
     // Code here to support video if enabled
     }
}
//write image(clean) to Album
- (void) saveImage
{
```

```
    UIImageWriteToSavedPhotosAlbum(imageView.image , nil, nil, nil);
}
```

行動應用 App 基本資安規範：4.1.2.2.敏感性資料利用

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：CSA MAST 4.2.2, ENISA SSDG1.5, NIST 800 SP-163 3.1.4

參考來源：

iOS：

- Location and Maps Programming Guide(查詢時間：2016/10/1)
- forensic-analysis-iOS-devices - sans.org(查詢時間：2016/10/1)
- github 5teev MetaPhotoSaveApp(查詢時間：2016/10/1)

CWE/OWASP：

- M4 - Unintended Data Leakage (查詢時間：2016/10/1)
- CWE 200(查詢時間：2016/10/1)

CPE-05:敏感性資料於頁面顯示時應依隱碼標準要求處理，如必要顯示明碼需先經使用身分認證及授權。

說明：

為保護使用者與資料安全，依個人資料保護法施行細則第 17 條所述：無從識別特定當事人，指個人資料以程式碼、匿名、隱藏部分資料或其他方式，無從辨識該特定個人者。故建議於頁面顯示時，必須以隱藏部分或程式碼方式處理。

隱碼標準可以參考 CNS 29191「資訊技術－安全技術－部分匿名及部分去連結鑑別之要求事項」國家標準或相關產業安全標準，如支付卡產業安全標準(PCI DSS)實作。

另外依照相關規定，如遇【交易確認頁】，可完全顯示個人資料明碼。

CNS 29191 提供個人資料部分匿名及部分去連結鑑別之框架，並建立其要求事項。重點內容如下：

- 為配合個人資料保護之施行，制定本項國家標準，提供各項資訊平台處理個人隱私資料保護之參考依據。
- 為確保在交易過程中，當事人保持匿名，並於交易行為無法透過資訊連結而得知個人資料，以確保個人隱私，本標準提供部分匿名及部分去連結鑑別之一般原則、框架、要求事項，並提供相關使用案例。
- 要求事項：
  - (a)宣稱者應由查證者鑑別，而不能由查證者識別 - 對於向查證者保持匿名之宣稱者，其交易不應提供任何可用 以識別宣稱者之資訊，但允許查證者證實宣稱者持有有效之信符
  - (b)鑑別紀錄單本身不得提供能連結同一宣稱者多項鑑別交易 之資訊 - 對於向查證者保持不可連結之宣稱者，其交易不得提供可連 結同一宣稱者履行之多項交易的任何資訊
  - (c)鑑別紀錄單應包含供指定開啟者重新識別宣稱者之必要資 訊 - 為使指定開啟者之後能重新識別宣稱者，成功交易產生之紀錄單應提供識別宣稱者之資訊。注意，於適當情況下，指定 開啟者可使用其他資訊以進行重新識別
  - (d)指定開啟者應能提供所宣稱身分為正確之證據 - 為避免指定開啟者之不誠實宣稱，指定開啟者應能提供重新 識別程序已正當履行之證據

CNS29191 可至國家標準(CNS)網路服務系統查詢。

如果要查閱明碼的敏感性資料，如完整姓名、信用卡號、通行密碼及連絡電話等，需實作身分認證後顯示。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

Android：Java

為符合個資法要求，當無法確定是否是使用者本人時，敏感性資料在顯示時，需要進行部分隱藏。若必需提供完整敏感性資料來進行作業時，開發人員需確定是使用者本人(例如：APP 程式要求使用者登入後台主機以進行身分驗證)，才可以明碼顯示之。

Android:*以下程式提供在設備畫面上呈現，隱藏第 456 碼的身分證字號*

```
TextView textView1 = (TextView) findViewByID(R.id.textView1);
     textView1.setText(UserId.substring(0, 3) + "***" + UserId.substring(3+3);
```

iOS：Objective-C

而為了取代敏感資料(例如：帳號/身分字號)，建議於資料庫產生另一個代號，可參考 Hash 雜湊值可以用於唯一標的識別的秘密資訊。經由將帳號改由另一個代號來處理，可降低原始帳號的顯示問題。產生代號時，也需注意代號重複問題，可透過索引與部分亂數的方法，以檢核並組成代號。

例如：SHA-1，可產生 160 位元的輸出；SHA-256，可產生 256 位元的輸出。

茲列出隱藏部分原始碼範例：

隱藏部分(隱藏身分字號之 456 碼)

```
    NSString *str = self.edit_input.text;
    self.label_Account.text = [str stringByReplacingCharactersInRange:NSMakeRange(3,
3)  withString:@"***"];
```

行動應用 App 基本資安規範：4.1.2.2.敏感性資料利用

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：NIST 800 SP-163 3.1.4

參考來源：

- 應用系統之資料遮罩技術實作分享，財金資訊公司賴威慎/許嘉家(查詢時間：2016/10/1)
- 個人資料保護法-全國法規資料庫(查詢時間：2016/10/1)
- Hash function - From Wikipedia(查詢時間：2016/10/1)
- SHA-1 - From Wikipedia(查詢時間：2016/10/1)

CPE-06:加密傳送敏感性字串查詢，並使用有 XSRF token 保護的 POST 指令發送，以防止重送攻擊(replay attack)。

說明：

若將敏感資料放入查詢參數，或者使用 GET 方式，這些資料的洩漏，都可能會被系統機制，記錄下來，或保存下來。例如：網頁查詢紀錄、網頁伺服器或代理器日誌等)。另一方面，使用未加密的查詢參數，也應該被避免。如果攻擊者在盜取這些資料後，就可能通過使用者漏洞資料來假冒權限。

使用安全的 POST 指令與 XSRF token 保護發送的使用者資料。POST 資料不會預設被記錄在可以被發現的查詢字元串資料區域。無論是 POST 或 GET，應使用的臨時交談 cookie。使用非零初始化向量和臨時交談密鑰進行資料加密的也可以幫助防止重放攻擊(Replay Attack)。如果需要的話，查詢字元串資料可以利用使用安全演算法，例如：Diffie-Hellman 主機之間協商臨時交談密鑰進行加密。加入序號欄位或隨機亂數並編碼驗證，也能防止攻擊。

由於 XSRF token 的功能是由主機提供，所以提供主機端程式碼範例。請參考 SERVER-07。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.2.敏感性資料利用

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：

Pinto, Marcus (2007). The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws (Kindle Locations 2813-2816). Wiley. Kindle Edition. (查詢時間：2016/10/1)

CWE/OWASP：

- M2 - Insecure Data Storage, M4 - Unintended Data Leakage(查詢時間：2016/10/1)
- CWE 598(查詢時間：2016/10/1)

CPE-07:行動應用 App 本地及伺服器端可使用代碼取代完整金融帳號或其他個人資料，如銀行帳戶及信用卡號。

說明：

行動支付為近來新興熱門議題，過去在實體世界信用卡交易需要有帶有磁條或是晶片金融卡的信用卡，經過商家銷售點讀卡機傳送至信用卡收單機構及發卡機構認證後，完成支付，但在行動支付或是線上支付就無法以實體卡片來支付，通常是直接在 App 或是網頁輸入信用卡號、有限期限及卡片安全碼(卡片背面磁條旁 3 位數字，VISA 稱為 CVV，即 Card Verification Value，而 Mastercard 稱為 Card Validation Code，CVC)，雖然便利但很容易因為實作儲存或傳輸不夠安全，在行動應用 App 本地端及伺服端或是在兩端之間傳輸時被竊取，造成信用卡資料外洩，降低這樣風險的其中一項策略是減少真實卡號在交易過程暫存或是通過不信任網路的機會，使用代碼(Tokens)是一項常用實務，Token 的概念起緣於「代幣」(Token coin)，在 1970 年代的數位風潮剛興起之時，「代碼化」(Tokenization)技術因應資料安全而生，用途是保護機密資訊不外洩，例如：銀行帳戶、醫療紀錄、犯罪紀錄、股票交易等。最常用於信用卡交易過程當中，就像是在信用卡之外延伸一層防護膜，這層防護膜就是「代碼化」。

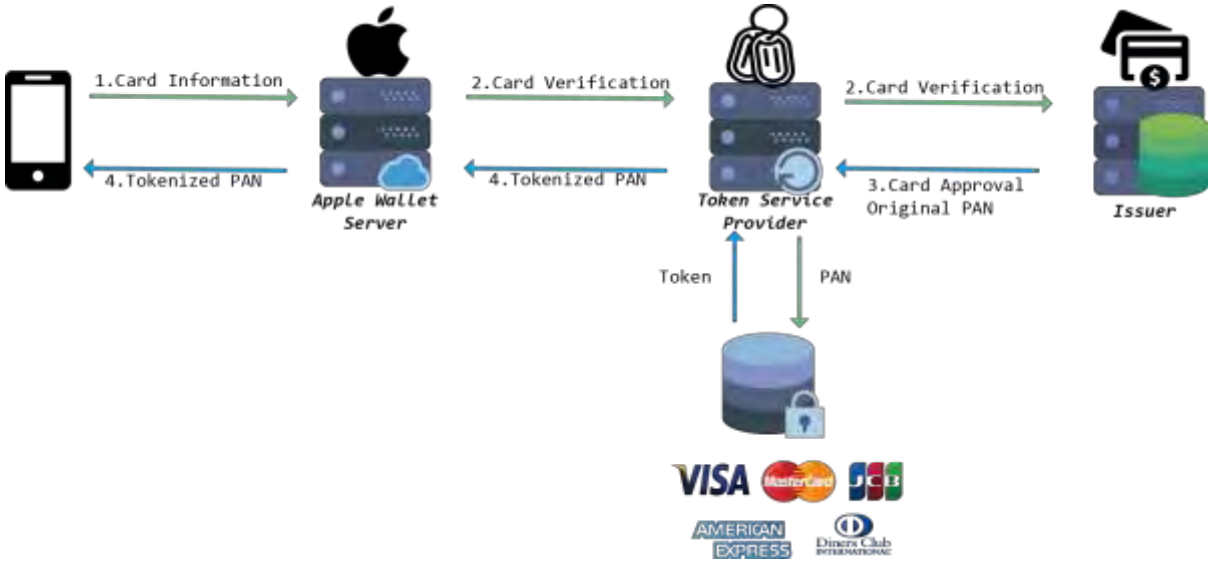
代碼化實作架構可參考支付卡產業安全標準由 PCI Security Standards Council 發布之 PCI DSS Tokenization Guidelines(查詢時間：2016/10/1)及 Tokenization Product Security Guidelines－Irreversible and Reversible Tokens(查詢時間：2016/10/1)。

代碼化系統捨棄了以卡片做為交易媒介的概念，改讓隨機產生的程式碼代替信用卡號來做交易，每個 Token 都是一組不重複且不具任何意義的序號，也無法被演算法反解回信用卡的資訊，每個程式碼也只能在特定時間和店家內使用，大大減低了機密資料外漏的風險。

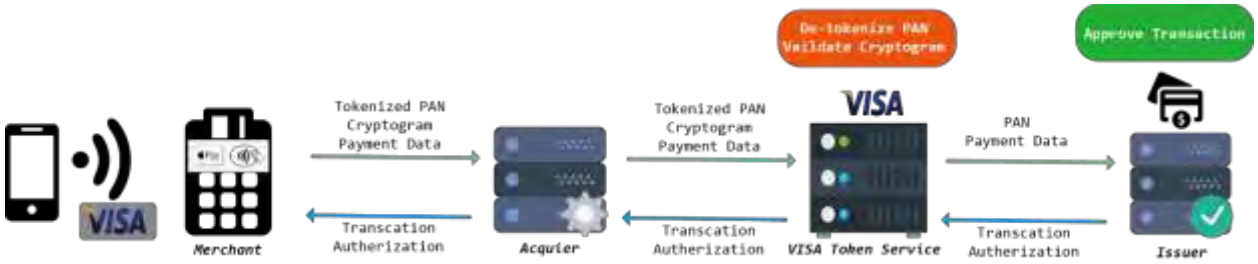
實際的使用機制是利用智慧型手機連結到 Token 服務系統，並將信用卡號替換成一組 Token，而店家(Merchant)從消費者那裡得到程式碼後，經過收單機構(Acquirer)和支付網路(Payment Network)傳送到發卡行(Issuer)，發卡行再透過代碼解碼(De-Token)還原該信用卡資訊、授權交易，並重新製碼(Re-Token)回傳給商家以確認交易，而整個過程只有發卡行與支付網路會知道消費者的信用卡資訊。

國際信用卡組織 Visa,Master 及各行動支付平台 Android Pay,Apple Pay,Samsung Pay 及台灣行動支付聯盟(twMP)都支援使用代碼化服務商(TSP)。以下以 Apple Pay (查詢時間：2016/10/1)為例:

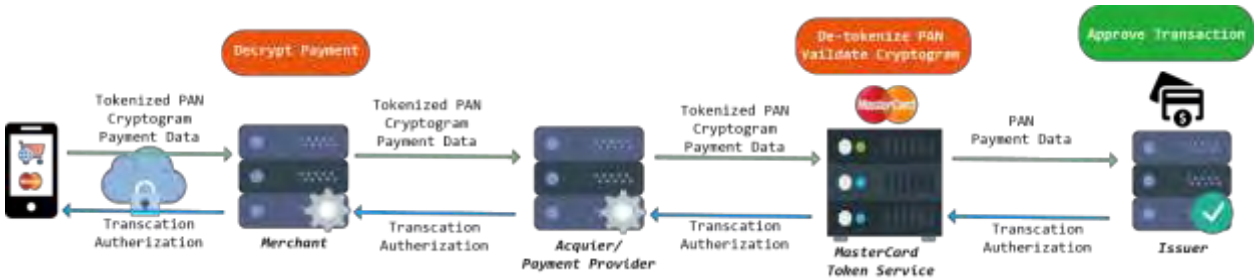
1. Apple Pay 在使用前需將實體信用卡（VISA,Master 等信用卡組織之銀行信用卡）加入 iOS 裝置，可以透過 iTune 關聯信用卡到 Passbook 或由 iSight 掃描，並經過 Apple、信用卡組織及發卡機構的三方認證，流程如下：



2. Apple Pay 的線下（Offline）支付流程：以在實體商家（Merchant）使用 iPhone 的 Apple Pay 消費為例，將已儲存在 iPhone 的 VISA 卡以 Tokennize PAN(代碼化支付卡帳號)及消費資訊，通過收單行認證，傳送至 VISA 的 Token Service 進行反解出真實 PAN 及交易資訊，傳送至發卡行（Issuer）進行核准，再反向傳送交易授權（Transcation Autherization）至實體商家完成交易，流程如下圖：



3. Apple Pay 的線上（Online）支付流程：以使用 MasterCard 信用卡線上購物為例，使用者於透過加密網路將支付 Tokennize PAN(代碼化支付卡帳號)及消費資訊傳送至線上商店，再由 Acquier/Payment Provider 傳送至 MasterCard 的 Token Service 進行反解出真實 PAN 及交易資訊，傳送至發卡行（Issuer）進行核准，再反向傳送交易授權（Transcation Autherization）至線上商家完成交易，流程如下圖：



安全開發生命週期：設計、開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.2.敏感性資料利用

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：

- PCI DSS Tokenization Guidelines(查詢時間：2016/10/1)
- Tokenization Product Security Guidelines－Irreversible and Reversible Tokens(查詢時間：2016/10/1)
- Apple Pay 離我們多遠(查詢時間：2016/10/1)

3.1.6. 敏感性資料儲存(F)

CPF-01:行動應用 App 如需要儲存敏感性資料需依 CPD-02 規劃於可信任之應用程式商店或行動應用程式內聲明。

說明：

行動應用 App 如需要儲存 1.5.3 定義敏感性資料參考 CPA-01 將行動應用 App 需要儲存敏感性資料於 Google Play 及 App Store 及行動應用 App 的隱私權政策中聲明。

安全開發生命週期：設計階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.3.敏感性資料儲存

行動應用 App 基本資安檢測基準：

4.1.2.3.1.(1)檢查行動應用程式是否於可信任之應用程式商店或行動應用程式內聲明。

各國開發要點參考：N/A

參考來源：N/A

CPF-02:行動應用 App 儲存敏感性資料應實作提示使用者同意及拒絕選項，提示使用者選擇時機可於(1)安裝時(2)當敏感資料被儲存或傳送前(3)預設設定為關閉同意，需使用者自行開啟。

說明：

行動應用 App 如需儲存敏感性資料，可參考 CPD-05 及 CPD-06，於 App 使用者介面實作彈出視窗提示使用者，選擇同意或拒絕選項，當使用者選擇同意後才能呼叫儲存 API，實作儲存請參考 CPF-03~CPF-14。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.3.敏感性資料儲存

行動應用 App 基本資安檢測基準：

4.1.2.3.1.(2) 檢查行動應用程式是否於可信任之應用程式商店或行動應用程式內取得使用者同意。

4.1.2.3.1.(3) 檢查在未聲明或未取得使用者同意敏感性資料儲存的情況下，行動應用程式是否未儲存敏感性資料。

4.1.2.3.2.(1) 檢查行動應用程式是否提供使用者拒絕儲存敏感性資料之選項。

4.1.2.3.2.(2) 檢查在使用者拒絕敏感性資料儲存的情況下，行動應用程式是否未儲存敏感性資料於行動裝置。

各國開發要點參考：ENISA SSDG 7.2 , JSSEC AASDCG 4.10

參考來源：N/A

CPF-03:行動應用 App 應依 CPD-03 規劃及實作適當資料儲存區域劃分，如記憶體、暫存檔、資料庫、日誌、全域可讀(公用)、不同敏感資料類別及內、外部儲存媒體，並依最小權限原則設定存取權限及安全機制，如加密、分割或代碼化。

說明：

行動應用 App 設計及開發實作人員應熟悉 Android 或 iOS 提供的內建資料儲存區域及權限控制方法，規劃實作適當資料儲存區域劃分，如記憶體、暫存檔、資料庫、日誌、全域可讀(公用)、不同敏感資料類別及內、外部儲存媒體，並依最小權限原則設定存取權限及安全機制，如加密、區隔或代碼化。

Android 的底層是基於 Linux 內核，提供了下列安全機制：

●內核安全：

Linux 內核的一個基本的安全目標是使用者資源彼此隔離。Linux 的安全理念是保護彼此使用者資源。因此 Linux 提供：

- 防止使用者 A 讀取使用者 B 的檔案。
- 為了保證使用者 A 不消耗使用者 B 的記憶。
- 為了保證使用者 A 不消耗使用者 B 的 CPU 資源。
- 為了保證使用者 A 不消耗使用者 B 的設備(如電話，GPS，藍牙)。

●應用程式沙箱(Application Sandbox)

Android 平台利用 Linux 的基於保護使用者目的，作為鑑定和分離的應用程式資源的一種手段。Android 系統分配一個唯一的使用者 ID(UID)每個 Android App 運行它作為一個該使用者單獨的程序(Processes)。這種方法不同於其他作業系統(包括傳統的 Linux 配置)，其中多個應用程式使用相同的使用者權限運行。

這將建立一個內核級應用程式沙箱。

內核在通過標準 Linux 的配置，例如：分配給應用程式的 User ID 和 Group ID 於 process 層級級強制應用程式和系統之間的安全性。預設情況下，App 間不能彼此互動，App 必須限制在作業系統存取。如果 App A 試圖做一些類似的惡意應用程式讀取 App B 的資料，或未經允許撥打電話(這是一個單獨的應用程式)，那麼作業系統可以防止這一點，因為 App A 沒有對應的使用者權限。沙箱是簡單且可稽核，並根據使用者將 processess 和 file permissions 分離。

一些作業系統，記憶體損壞錯誤通常導致完全危及設備的安全性。在 Android 的環境下，由於所有 App 及其資源在作業系統層級被沙箱化。記憶體損壞錯誤將只允許執行任意程式碼在特定 App 的環境與作業系統建立的權限。

像所有的安全功能，應用程式沙箱並非牢不可破。然而，在一個正確配置的設備建立的應用程式沙箱，必須與 Linux 內核的安全性妥協。

●系統分區和安全模式

該系統分區包含 Android 的內核以及作業系統的函式庫，應用程式執行階段(Application runtime)，應用程式框架(Application Freamwork)和應用程

式。這個分區設置為唯讀(Read Only)。當使用者啟動設備進入安全模式，第三方App 可以由設備擁有的使用者手動啟動，非由作業系統自動帶起來。

●**文件系統權限**

在 UNIX 風格的環境中，文件系統權限確保一個使用者不能改變或者閱讀其他使用者的文件。在 Android 環境的情況下，每個 App 運行在擁有它自己的使用者環境中。除非開發人員明顯地暴露文件到其他 App，原則上由一個 App 所創建的文件不能被讀出或由另一個 App 改變。

●**安全增強型 Linux**

Android 使用安全增強型 Linux(SELinux)的應用存取控制策略，建立強制存取控制(MAC)的環境。詳細資訊請參考驗證安全增強型 Linux 的 Android(查詢時間：2016/10/1)

●**加密**

Android 提供了一套可由應用程式使用加密的 API。這些包括標準的和常用的加密演算法，例如：AES，RSA，DSA 和 SHA 的實現。此外，提供了用於更高等級的協議，如 SSL 和 HTTPS 的 API。

Android 4.0 的推出了鑰匙串(Key Chain)(查詢時間：2016/10/1)class，允許應用程式儲存私鑰和證書鏈系統憑證在 Key Chain。

在 Android 平台上讀寫檔案的也是利用 Java 的 File、InputStream 以及 OutputStream 物件來達成。Android 系統對 App 的使用空間與檔案操作係透過系統提供的 Context 與 Environment 物件可以讓開發人員快速的進行檔案的各種操作。詳細請參考 Android 平台的檔案讀寫方式(查詢時間：2016/10/1)

iOS 提供了下列安全機制：

在 iOS 上的檔案資料保護，除了 iOS 裝置內建的硬體加密功能，Apple 也使用名為「資料保護」的技術，來進一步保護裝置上快閃記憶體中儲存的資料。「資料保護」可讓裝置回應如來電之類的常見事件，也可以對使用者資料啟用較高層次的加密。「訊息」、「郵件」、「行事曆」、「聯絡資訊」、「照片」和「健康」資料值等主要系統 App 預設都會使用「資料保護」，而安裝於 iOS 7 或更新版本上的第三方 App 可自動獲得此項保護措施。

「資料保護」是透過建構和管理密鑰階層來完成導入，並建立在每部 iOS 裝置的硬體加密技術上。「資料保護」藉由將每個檔案指定給某個類別，進而對檔案逐一進行控制，可取用性則取決於該類密鑰是否已解鎖。

●**架構概覽**

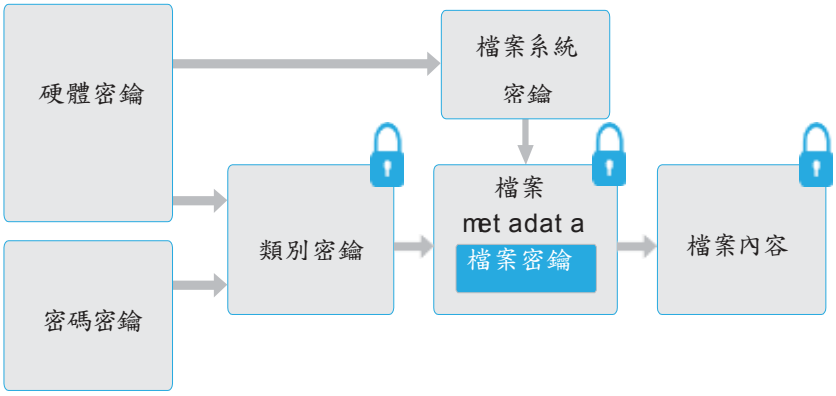
每次在資料分割區上製作檔案時，「資料保護」都會製作一個新的 256 位元密鑰(「檔案專屬」密鑰)，並將其提供給硬體 AES 引擎，此引擎會使用該密鑰採用 AES CBC 模式對寫入快閃記憶體的檔案進行加密。(在配備 A8 處理器的裝置上，會使用 AES-XTS)初始化向量(IV)使用檔案的區塊偏移量進行計算，使用檔案專屬密鑰的 SHA-1 雜湊進行加密。

依據每個檔案的可取用性情況，檔案專屬密鑰會使用其中一個類別密鑰進行封裝。就像所有其他封裝一樣，這是使用 NIST AES 密鑰封裝、依據 RFC 3394 來執行。封裝的檔案專屬密鑰會儲存在檔案的 metadata 中。

當打開檔案時，系統會使用檔案系統密鑰來解密其 metadata，以呈現封裝的檔案專屬密鑰以及表示其保護類別的記號。檔案專屬密鑰會使用類別密鑰來解除封裝，然後提供給硬體 AES 引擎，該引擎會在從快閃記憶體中讀取檔案時，對檔案進行解密。

所有封裝檔案密鑰的處理作業會在 Secure Enclave 中進行；檔案密鑰永遠不會直接提供給應用程式處理器。在開機時，Secure Enclave 會與 AES 引擎進行協調以獲取臨時密鑰，當 Secure Enclave 解除封裝檔案密鑰時，會透過臨時密鑰來重新封裝，並傳送回應用程式處理器。

檔案系統中所有檔案的 metadata 都使用隨機密鑰進行加密，該密鑰是在首次安裝



iOS 或使用者清除裝置時製作而成。檔案系統密鑰則儲存在 Effaceable Storage 中。因為該密鑰儲存在裝置上，因此它不是用來維護資料的機密性，而是可以視需求快速清除，(由使用者從「清除所有內容和設定」選項來清除，或者由使用者或管理者從行動裝置管理(MDM)伺服器、Exchange ActiveSync 或 iCloud 發出遠端清除指令來清除)。以此方式清除密鑰將會透過加密的方式讓裝置上的所有檔案無法取用。

檔案的內容使用檔案專屬密鑰進行加密，該密鑰使用類別密鑰封裝並儲存在檔案的 metadata 中，檔案 metadata 接著又使用檔案系統密鑰進行加密。類別密鑰使用硬體 UID 取得保護，而某些類別則透過使用者密碼取得保護。此階層架構同時提供了彈性與效能。例如：更改檔案的類別只需要重新封裝其檔案專屬密鑰，更改密碼只需要重新封裝類別密鑰。

●資料保護類別

在 iOS 裝置上製作新檔案時，用來製作的 App 會替檔案指定一個類別。每個類別使用不同的規則來決定資料何時可供取用。基本類別和規則會在下面各點說明。

●完整保護

(NSFileProtectionComplete)：類別密鑰會使用從使用者密碼和裝置 UID 所衍生的密鑰加以保護。使用者鎖定裝置後不久(若「需要密碼」設定為「立即」時，則為

10 秒)，已解密的類別密鑰便會被捨棄，讓此類別中的所有資料只有在使用者再次輸入密碼或使用 Touch ID 解鎖裝置時，才可以取用。

●未打開檔案的保護

(NSFileProtectionCompleteUnlessOpen)：有些檔案可能需要在裝置鎖定時寫入。其中一個不錯的例子是電子郵件的附件在背景下載。此行為是藉由使用非對稱橢圓曲線加密技術(Curve25519 的 ECDH)來達成。常見的檔案專屬密鑰則是使用 One-Pass Diffie-Hellman Key Agreement(如 NIST SP 800-56A 中所述)所衍生的密鑰加以保護。

該協議的臨時公用密鑰與封裝的檔案專屬密鑰一起儲存。KDF 是鏈結密鑰衍生函數(Approved Alternative 1)，如 5.8.1 of NIST SP 800-56A 中所述。AlgorithmID 已忽略。PartyUInfo 和 PartyVInfo 分別是臨時靜態公用密鑰。SHA-256 則用於雜湊函數。檔案一旦關閉，檔案專屬密鑰便會從記憶體中清除。若要再次打開檔案，系統會使用「未打開檔案的保護」類別的專用密鑰和檔案的臨時公用密鑰來重新製作共享密鑰；其雜湊會用來解除封裝檔案專屬密鑰，該密鑰接著會被用來解密檔案。

●首次使用者認證前的保護

(NSFileProtectionCompleteUntilFirstUserAuthentication)：此類別與「完整保護」類別的行為方式相同，只是在鎖定裝置時，已解密的類別密鑰不會從記憶體中移除。此類別中的保護和桌上型電腦完整卷宗的加密有類似的屬性，可防止資料因重新啟動而遭到攻擊。對於未指定至「資料保護」類別的所有第三方 App 資料，這是預設類別。

●無保護

(NSFileProtectionNone)：此類別密鑰僅受到 UID 的保護，並且儲存在 Effaceable Storage 中。因為解密該類別中之檔案所需的所有密鑰都儲存在裝置上，因此加密的唯一好處就是可以進行快速的遠端清除。即使未對檔案指定「資料保護」類別，此檔案仍會以加密形式儲存(就像 iOS 裝置上的所有資料一樣)。

●鑰匙圈(Keychain)資料保護

許多 App 需要處理密碼和其他簡短但較為敏感的資料，如密鑰和登入 Token。iOS 鑰匙圈提供了儲存這些項目的安全方式。

鑰匙圈是以儲存在檔案系統中的 SQLite 資料庫的方式導入。只有一個資料庫；securityd 服務程式會決定哪些鑰匙圈項目可被每個處理程序或 App 取用。鑰匙圈取用 API 會產生對服務程式的呼叫，進而查詢 App 的「keychain-access-groups」、「application-identifier」和「application-group」權限。取用群組允許在 App 間共享鑰匙圈項目，而非將取用權限限制為單一處理程序。

鑰匙圈項目只能在來自同一開發人員的 App 間共享。管理方式是要求第三方 App 使用取用群組，並使用透過 Apple Developer Program(Apple 開發人員計畫)或透過應用程式群組來為其分配前置碼。對前置碼的要求和應用程式群組唯一性，是透

過程式碼簽署、佈建描述檔 Apple Developer Program(Apple 開發人員計畫)強制執行。

系統用來保護鑰匙圈項目的類別結構，與檔案「資料保護」中使用的類別結構相似。這些類別具有與檔案「資料保護」類別相同的行為，但使用的密鑰不同，所屬 API 的名稱也不同。

●**Keybag**

檔案和鑰匙圈「資料保護」類別的密鑰會蒐集在 Keybag 中加以管理。iOS 使用五種 Keybag：使用者、裝置、備份、託管和「iCloud 備份」。

安全開發生命週期：設計、開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.3.敏感性資料儲存

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：CSA MAST 4.2.7.

參考來源：

Android：

- 系統和內核安全(查詢時間：2016/10/1)
- Android 平台的檔案讀寫方式(查詢時間：2016/10/1)

iOS：iOS Security Guide(查詢時間：2016/10/1)

CPF-04:在考量行動智慧裝置本地(Device)端與 Server 端相對安全性，將敏感性資料優先儲存於 Server 取代儲存於 Device 本地端，但前提是安全連線機制已建立及 Server 端儲存的安全機制足夠或優於 Device 端。

說明：

行動應用 App 儲存敏感性資料之落地位置，就 Device 本地端與 Server 伺服器就所在環境實體安全風險，經常暴露在公眾環境或可能存取不明 WiFi、藍芽或 NFC，被惡意攻擊者竊取或近端資料劫奪安全風險，明顯高於放置於實體環境企業或資料中心的 Server 端為高，相較之下敏感性資料以放在 Server 端為宜。

但前提是敏感性資料自 Device 端向 Server 端傳送通道是加密，且 Server 的安全防護水準也需明顯優於 Device 端。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.3.敏感性資料儲存

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 1.2

參考來源：N/A

CPF-05:優先使用行動裝置作業系統提供的 API 實作儲存加密機制，加密模組應通過 FIPS 140-2 認證，使用加密/簽章演算法與金鑰長度與如下：

(1)對稱式加密，金鑰有效長度為 128 位元、192 位元或 256 位元之先進加密標準(AES)演算法。

(2)對稱式加密，金鑰有效長度為 112 位元或 168 位元之三重資料加密演算法(Triple DES)演算法。

(3)非對稱加密，金鑰有效長度為 1,024 位元以上的 RSA 或 160 位元以上的 ECC 加密演算法。

(4)簽章雜湊函數/金鑰長度 SHA-256

說明：

敏感性資料建議不應寫入檔案。若有寫入必要性，也須做好安全分類與加密保護。建議使用系統上提供之公認安全的系統加密函數，不要隨意使用第三方資料保護，以免強度不足時，反而會降低安全保護。當使用系統加密函數時，也需仔細了解該功能的用途、參數與限制。

iOS 行動智慧裝置都配備專屬的 AES 256 加密引擎，其內建於快閃儲存空間與主系統記憶體間的 DMA 路徑中，可讓檔案加密具備高度效率。裝置的唯一識別碼(UID)與裝置群組識別碼(GID)是 AES 256 位元的密鑰，該密鑰已在製作過程中融入(UID)或編譯(GID)到應用程式處理器和 Secure Enclave 中。Secure Enclave 的 UID 和 GID 只能由 Secure Enclave 專屬的 AES 引擎使用。每部裝置的 UID 都是唯一的，Apple 或其他供應商都不會有所記錄。除了 iOS 裝置內建的硬體加密功能，Apple 也使用名為「資料保護」的技術，來進一步保護裝置上快閃記憶體中儲存的資料。「資料保護」說明可參考 CPF-03。

iOS 原生沒有支援 RSA 加密，可透過 iOS 對 Security.framework 的封裝，實現 RSA 非對稱加密解密。可獲取公開金鑰和私密金鑰，可對資料進行公開金鑰和私密金鑰的加解密。這種方式需要對密匙對進行處理，根據 public key 生成證書，通過 private key 生成 p12 格式的密鑰，實作可參考 iOS 開發探索-RSA 加解密(查詢時間：2016/10/1)。

因為 Android 是開放系統平台運行於各家廠商不同硬體，無法像 iOS 有專屬硬體安全功能可以實作，Android 在作業系統提供了支援 TEE(Trusted Execution Environment)Trust TEE 的軟體元件及 API，提供全碟加解密及數位版權管理(Digital Right Management,DRM)來保護敏感性資料，Android 的全磁盤加密是基於 dm-crypt，這是一個內建核心的功能，在塊設備層(block device layer)工作。嵌入式多媒體卡(Embedded MultiMediaCard，eMMC)對內核呈現自己為類似塊設備的閃存設備。正因為如此，不可能在 YAFFS (Yet Another Flash File System)檔案系統加密，而會直接運用原始 NAND 閃存晶片進行加密。

加密算法是 128 的高級加密標準(AES)與密碼塊鏈接(CBC)，並實作 ESSIV(使用 SHA256 雜湊演算法)。主密鑰是通過呼叫 OpenSSL 庫採用 128 位元 AES 加密。必須使用 128 位元以上的密鑰(可選 256 位元)。

Android 官方亦無 RSA 的加解密方法，實作可參考「Android RSA 與 Java RSA 加密不同標準產生問題的解決方法」（陳小默的博客）」(查詢時間：2016/10/1)。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.3.敏感性資料儲存

行動應用 App 基本資安檢測基準：

4.1.2.3.5.(1) 檢查行動應用程式是否採用金鑰有效長度為 128 位元、192 位元或 256 位元之先進加密標準(AES)。

4.1.2.3.5.(2) 檢查行動應用程式是否採用金鑰有效長度為 112 位元或 168 位元之三重資料加密演算法(Triple DES)。

4.1.2.3.5.(3) 檢查行動應用程式是否採用 SHA-256。

各國開發要點參考：NIST 800 SP-163 3.1.4b,ENISA SSDG 1.3, JSSEC AASDCG 5.6

參考來源

Android：Full Disk Encryption(查詢時間：2016/10/1)

iOS：iOS security Guide P.10(查詢時間：2016/10/1)

CWE/OWASP：

M2 - Insecure Data Storage(查詢時間：2016/10/1)

CWE 312, 313, 522, 215(查詢時間：2016/10/1)

CPF-06:實作從加密儲存區域解密資料，需經使用者或系統管理員身分認證及授權始可運作。

說明：

如需由加密儲存資料解密資料，參考 CPF-03 Android 應實作經身分認證後始能呼叫解密，Android 的全碟加解密請參考如何實作 Android 的全磁盤加密(查詢時間：2016/10/1)。

在 iOS 裝置上製作新檔案時，用來製作的 App 會替檔案指定一個類別。每個類別使用不同的規則來決定資料何時可供取用。iOS 有「完整保護」、「未打開檔案」、「首次使用者認證前的保護」及「無保護」等 4 種資料類別，以「完整保護」(NSFileProtectionComplete)類別為例，使用者再次輸入密碼或使用 Touch ID 解鎖裝置時才可以取用。「資料類別」保護機制詳見 iOS security Guide P.10(查詢時間：2016/10/1)。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.3.敏感性資料儲存

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：YD/T 2407-2013 5.6.3

參考來源：

Android：如何實作 Android 的全磁盤加密(查詢時間：2016/10/1)。

iOS：iOS security Guide P.10(查詢時間：2016/10/1)

CPF-07:敏感性資料或包含敏感性資料的日誌檔，除非已加密，應避免儲存於與其他行動應用 App 共用或全域可讀寫儲存區域或外部儲存媒體。

說明：

其他行動應用 App 共用或全域可讀寫儲存區域或外部儲存媒體均是未設任何權限且明文儲存資料區域，將敏感性資料或相關日誌儲存前述區域極易造成敏感性資料外洩。Android 預設是將資料儲存於行動應用 App 的私有儲存區域，其他的 App 無法存取它們，當 App 被卸載時會被一併刪除，呼叫 openFileOutput()與該文件的名稱和操作模式，MODE_PRIVATE 將創建文件，並使其專用於您的應用程式。其他可用的模式有：MODE_APPEND，MODE_WORLD_READABLE，和 MODE_WORLD_WRITEABLE。

注意：參數 MODE_WORLD_READABLE 和 MODE_WORLD_WRITEABLE 自 API 等級 17 以後已不能用於 Android N，使用它們開始將導致 SecurityException 錯誤訊息被拋出。這意味著 Android N 和更高版本不能通過檔案名稱共享的私有檔案，並試圖共享一個"file：//"URI 將導致 FileUriExposedException 錯誤訊息被拋出。如果您的應用程式需要與其他應用程式共享的私有檔案，它可以使用 FileProvider 在 FLAG_GRANT_READ_URI_PERMISSION 申請權限。另請參閱共享檔案。

iOS 現階段並無法將沙箱的私有資料像 Android 儲存在沙箱之外在全域可讀寫的區域。

安全開發生命週期：開發實作階段

不安全程式碼範例：Android：Java

```
• 在 API Level 17 Android 4.2 (JELLY_BEAN_MR1)之前的版本，可以允許開發人員的 APP 將資料輸出到 APP 公有區中，下列範例即為將資料輸出到公有區中，並設其權限為共同寫入。
  String FILENAME = "hello_file";
  String string = "hello world!";

  FileOutputStream fos = openFileOutput(FILENAME, Context.MODE_WORLD_READABLE,

• );
  fos.write(string.getBytes());
  fos.close();
```

安全程式碼範例：Android：Java

```
• 下列範例為安全的範列，此範例改將資料輸出到私有區中，並設其權限為私有，以進一步確保資訊安全。
  String FILENAME = "hello_file";
  String string = "hello world!";
```

```
FileOutputStream fos = openFileOutput(FILENAME, Context.MODE_PRIVATE);
fos.write(string.getBytes());
fos.close();
```

行動應用 App 基本資安規範：4.1.2.3.敏感性資料儲存

行動應用 App 基本資安檢測基準：

4.1.2.3.6.檢查行動應用程式是否儲存敏感性資料於其他行動應用程式預設無法存取之區域。

各國開發要點參考：ENISA SSDG 1.4,1.7, NIST 800 SP-163 3.1.3, 3.1.5

參考來源：Android：Storage Options(查詢時間：2016/10/1)

CPF-08:需注意記憶體暫存的敏感性資料，如固定金鑰與密碼的安全性，當不需要時或於一定合理期間應強制清除或使用於一定期間就失效的可變量金鑰替代。

說明：

在行動應用 App 使用時，使用者或應用程式會將特定資料儲存在記憶體中，以方便使用者離開或逾時之後，下次能夠直接使用。在 Android 上因為應用程式使用後停留在記憶體中，會被駭客利用除錯器竊取。另一方面，針對敏感的密鑰、密碼，建議不要用字串儲存，改用 Byte Array 的方式儲存，以免容易被發現。而使用過之後，也建議應該儘快清除，或確定會被系統記憶體回收機制回收。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

Android：Java

因為 Java 的 Garbage Collection(GC)的特性是，開發人員無法確認何時資料會真的被清空，而 String 是 immutable，基本上無法主動清空，就算指向 null，還是需要等待 GC 完成後，它才不會存在於記憶體中，最有效的方式為改以 byte array 來存放，在使用完之後，直接將每一個 byte 值覆寫為 0，如此一來立刻就能將資料從記憶體中清空，防止惡意人士從記憶體中盜取。

```
• SecretKey key = KeyGenerator.getInstance("DES").generateKey();
  byte[] data = key.getEncoded();//處理完相關作業之後立即清空，以避免存在於 RAM

for (byte oneByte:data){     oneByte=0;
}
```

iOS：Objective-C

例如：extern unsigned char * CC_SHA256(const void *data, CC_LONG len, unsigned char *md); 中的參數 data 來源，除了標準的資料輸入以外，還可加入長度、固定的 Salt 與私密的計算。例如：由 byte array 而來的 hash，或者計算後的 XOR。

```
• unsigned char obfuscatedSecretKey[] = { 0x33, 0xAD, 0xBE, 0xEF, 0xDE, 0xAD,
  0xBE, 0xEF, 0xDE, 0xAD, 0xBE, 0xEF, 0xDE, 0xAD, 0xBE, 0xEF };
  int hash = [Server ComputeHash:obfuscatedSecretKey];
  //One-at-a-Time Hash
  +(int) ComputeHash: (unsigned char*) data;
  {
     const int p = 16777619;
     int hash = (int)2166136261;
     int len = (int)strlen((char*)data);
```

```
    for (int i = 0; i < len; i++)
        hash = (hash ^ data[i]) * p;
    hash += hash << 13;
    hash ^= hash >> 7;
    hash += hash << 3;
    hash ^= hash >> 17;
    hash += hash << 5;
    return hash;
}
    // XOR the class name against the obfuscated key, to form the real key.
    unsigned char actualSecretKey[sizeof(obfuscatedSecretKey)];
    for (int i=0; i<sizeof(obfuscatedSecretKey); i++) {
        actualSecretKey[i] = obfuscatedSecretKey[i] ^ obfuscatedSecretKey[i];
    }
```

行動應用 App 基本資安規範：4.1.2.3. 敏感性資料儲存

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：

iOS：

http://www.splinter.com.au/2014/09/16/storing-secret-keys/ (查詢時間：2016/10/1)

http://burtleburtle.net/bob/hash/doobs.html(查詢時間：2016/10/1)

CWE/OWASP：

M4 - Unintended Data Leakage(查詢時間：2016/10/1)

CWE 316, 200(查詢時間：2016/10/1)

CPF-09:以設定或程式編寫關閉行動應用 App 本地端及網頁伺服器端暫存檔 HTTPS 流量資料，如日誌/除錯文件、Cookie、明文密碼、密碼 hash 值、網頁歷史、網頁暫存檔及屬性列表等。

說明：

開發人員經常忽略一些可以暫存資料，包括日誌/除錯文件、Cookie、網頁歷史、網頁暫存檔、屬性列表及文件等。為了防止資料被盜取，關閉任何資料儲存、資料暫存檔。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

Android：Java

```
此範例為設定如何停用 HttpURLConnection 的 cache。
URL url = new URL(urlString); HttpURLConnection connection = (HttpURLConnection)
url.openConnection();  connection.setDefaultUseCaches(false);
connection.setUseCaches(false);
```

iOS：Objective-C

手機端：清除連線 cache

```
• [[NSURLCache sharedURLCache] removeAllCachedResponses];
•    [[NSURLCache sharedURLCache] setDiskCapacity:0];
•    [[NSURLCache sharedURLCache] setMemoryCapacity:0];
```

主機端：.NET C#

```
//disable caching of Controller
[OutputCacheAttribute(VaryByParam = "*", Duration = 0, NoStore = true)] // will be
applied to all actions in MyController, unless those actions override with their own
decoration
public class MyController : Controller
{
  // ...
}
//disable caching for specific one
public class MyController : Controller
{
    [OutputCacheAttribute(VaryByParam = "*", Duration = 0, NoStore = true)] // will
disable caching for Index only
    public ActionResult Index()
    {
```

```
        return View();
    }
}
```

Server

瀏覽器(網頁內容/html)：

```
<meta http-equiv="Cache-Control" content="no-cache, no-store, must-revalidate" />
<meta http-equiv="Pragma" content="no-cache" />
<meta http-equiv="Expires" content="0" />
```

行動應用 App 基本資安規範：4.1.2.3.敏感性資料儲存

行動應用 App 基本資安檢測基準：

4.1.2.3.4.(1) 檢查行動應用程式是否未檢出將敏感性資料儲存於網頁暫存檔或自定義暫存檔。

各國開發要點參考：N/A

參考來源：

iOS： NSURLCache Class Reference - Apple developer (查詢時間：2016/10/1)

主機：OutputCache - .NET(查詢時間：2016/10/1)

網頁內容：Disable browser caching with meta HTML tags(查詢時間：2016/10/1)

CPF-10:行動應用 App 正式產品版本應關閉或移除除錯日誌，或避免儲存敏感性資料於未加密日誌儲存區。

說明：

開發人員應該考慮到除錯日誌可以在正式產品版本發布後帶來的風險。尤其是敏感性資料，一般我們建議他們 App 正式產品版本中禁用。通常使用的應用程式輸出的除錯資訊的 Android 系統日誌保存在儲存器中幾 KB 的循環緩衝區。這也可能是可以從內核意外中止的情況下恢復文件系統除錯日誌。

在設備重啟被清除，但在那之前任何與 READ_LOGS 權限的 Android 應用程式可以查詢的日誌。在最新的 Andoid 版本的日誌文件已更仔細地分離出來，不需要被請求的系統級權限。在 Android 中可以利用的 ProGuard 或 DexGuard 徹底刪除，呼叫在正式發布產品版本的 Log Class 方法，從而禁止呼叫所有 Log.d，Log.i，Log.v，Log.e 的方法。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

Android：Java

如開發人員不使用 ProGuard 或 DexGuard 來統一控制 Log 的行為，本指引提供此範例讓使用者可重新打包原的 android.util.Log，來達到統一控制 Log 行為

```
//自訂一個工具程式 Log, 將原有的 android.util.Log 重新包裝
public class Log{
//集中設定 FLAG 的狀態，如為開發環境則設為 true, 反之則為 false
    private static boolean FLAG = false;
    public static void d(String enable_tag, String message,Object...args){
        if(FLAG)
            android.util.Log.d(enable_tag, message+args);
    }
    public static void e(String enable_tag, String message,Object...args){
        if(FLAG)
            android.util.Log.e(enable_tag, message+args);
    }
    public static void v(String enable_tag, String message,Object...args){
        if(FLAG)
            android.util.Log.v(enable_tag, message+args);
    }
    public static void i(String enable_tag, String message,Object...args){
        if(FLAG)
            android.util.Log.i(enable_tag, message+args);
    }
}
```

iOS：Objective-C

```
• 於Release版本時，覆寫NSLog函式，以達到不洩漏資訊的目的
• NSLog rewrite when publishing
• //disable NSLog when Release
• #ifndef DEBUG
•  #define NSLog(...)
#endif
```

行動應用 App 基本資安規範：4.1.2.3.敏感性資料儲存

行動應用 App 基本資安檢測基準：

4.1.2.3.4.(2) 檢查行動應用程式是否未檢出將敏感性資料儲存於系統日誌或自定義日誌。

各國開發要點參考：ENISA SSDG 2.9,ENISA SSDG 7.5

參考來源：[psa disable nslog in your release builds(查詢時間：2016/10/1)]

CPF-11:行動應用 App 正式版本避免產生或傳送明文意外中止錯誤日誌。

說明：

如果一個應用程式意外中止錯誤(Crash)時，產生的輸出檔案是明文，可能提供給攻擊者有價值的資訊。於是確保應用程式有徹底的測試，以避免產生意外中止錯誤；良好的調整方法為移除中斷程式的函數。例如：Assert 這類的函數。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

Android：Java

```
本範例提供一重新打包的 Assert，來達成中央集中設定的行為
//自訂一個工具程式 Assert, 將原有的 juint.framework.Assert 中會使用到的方法，重新包裝
public class Assert{
//集中設定 FLAG 的狀態，如為開發環境則設為 true, 反之則為 false
    private static boolean FLAG = false;
    public static void assertEquals(int expected, int actual){
        if(FLAG) junit,framework.Assert.assertEquals (expected, actual);
    }
    public static void assertSame(Object expected, Object actual){
        if(FLAG) junit,framework.Assert.assertSame (expected, actual);
    }

}
```

iOS：Objective-C

在 iOS 中，NSAssert 斷言函式，可協助開發人員進行檢查程式的運作是否正確。XCode 的編譯設定，也預設將 NSAssert 函式在 Release 時，自動關閉其運作。


```
//也可加上 try catch 以免執行部分產生例外，而造成 crash dump.
//using try catch exception block
@try {
       //some danger block of codes
   }
   @catch (NSException *ex) {
```

```
        NSLog(@"Exception happened: %@", [ex description]);
    }
```

行動應用 App 基本資安規範：4.1.2.3.敏感性資料儲存

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：

iOS：

Handling Exceptions - apple developer(查詢時間：2016/10/1)

http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html(查詢時間：2016/10/1)

CWE/OWASP：

M2 - Insecure Data Storage, M4 - Unintended Data Leakage(查詢時間：2016/10/1)

CWE 312, 313, 522, 200(查詢時間：2016/10/1)

CPF-12:避免於行動應用 App 之程式碼、二進位碼或其他封裝程式中，直接嵌入密碼、身分驗證資訊或對稱式加解密演算法之密鑰(Keys)、初始化向量(Initial Vector, IV)。

說明：

行動應用 App 下載後，很可能會被惡意攻擊者以逆向工程進行研究，如果直接嵌入密碼、身分驗證資訊或對稱式加解密演算法之密鑰(Keys)、初始化向量(Initial Vector, IV)等加密因子，極易造成加密被破解，造成敏感性資料外洩。

實作加密應依優先使用行動應用平台內建安全機制，呼叫原生加密 API，密鑰管理（Key Management）由系統內核進行處理，無需另存金鑰於程式碼或其他易被破解儲存空間，內建如Android 的 Hardware-backed Keystore(查詢時間：2016/10/1)及 iOS 的 Secure Enclave(查詢時間：2016/10/1)。

如需在行動智慧裝置本地端儲存金鑰安全設計，建議參考下列做法（金融產業為必要，其餘產業可參考選用）：

(一)應採用下列任一技術保護金鑰：

1.採用晶片安全設計者，金鑰應儲存於符合我國國家標準 CNS 15408 EAL5、共通準則(Common Criteria)ISO/IEC15408 V2.3 EAL 5 或 FIPS 140-1 LEVEL 2(含)以上或其他安全強度相同之安全元件(SE)內，並能防堵市面上常見之攻擊破解方法。

以行動銀行為範例，目前尚無硬體安全元件應用單元。既使協作模組加入行動銀行擴充應用，採用硬體安全元件作為交易安控載具者(例如：行動支付)，其硬體元件之服務供應商都已配合上開金鑰保護規範。

3.採用軟體保護技術(如白箱加密法並配程式碼混淆技術)。

目前行動銀行最常被採用涉及金鑰之應用技術包含但不限於 RSA 非對稱式加密演算法。實務上來說，私密金鑰(包含但不限於 RSA 非對稱式加密演算法)配置於用戶端之管理辦法建議如下。

(1)以身分驗證機制於適當期間配置私密金鑰於用戶端，但需管理生命週期。

(2)私密金鑰暫時儲存於用戶端亦需搭配適當加密演算法，讓私密金鑰存取保持一定程度之安全性。

(3)私密金鑰之產生可以藉由多維變數演算。必要時，結合每一次動態交易資訊，驗證私密金鑰之可靠度。

白箱加密法技術性做法建議如下：

(1)Android 應用程式取用 AES 函數，AES Key 使用 Device ID、Package Name、Application Version 與 URL Domain 四維變數進行邏輯運算，取出 32 位元組字串作為密鑰。

(2)iOS 應用程式取用 AES 函數，iOS Keychain 使用 Device ID、Package Name、Application Version 與 URL Domain 四維變數進行邏輯運算，取出 32 位元組字串作為密鑰。

4.經第三方機構確認其安全防護。

例如：協力廠商所提供之掃描程序。不建議採用此方法。

(二)透過金鑰運算(如 OTP、TAC 等)應用於非約定轉入帳戶之轉帳交易，應確認金鑰儲存於客戶指定之行動裝置。

1.若為硬體安控元件應用(憑證應用)，註冊時期已綁定行動裝置資訊。

2.若為軟體安控元件應用，下列注意事項二擇其一。

(1)註冊時期可以協作裝置綁定服務結合交易安控確認。

(2)私密金鑰之產生結合行動裝置唯一資訊，於提交應用時，用戶端或伺服器經由演算程序得知行動裝置之合法性。

(三)應於交易時增設存取控管或人工確認，限制由可信任行動應用程式存取，以防止遭受惡意程式發動阻斷服務攻擊或偽冒交易。

1.程序存取控管：

目前行動銀行最常被採用方法係透過通訊介面設計辨識交易來源之合法性。辨識交易來源可以針對結合提取行動裝置資訊，經過一定程序之演算程序，封裝提交伺服端，由伺服端再經過一定程序之演算程序再辨識通訊來源之合法性。常見之演算程序包含加密協定、雜湊函數或押碼等。

2 人工附加確認(建議採用本項)：

(1)若為數位憑證應用：採用輸入憑證密碼確認。

(2)若為私密金鑰應用：採用對話方塊或文按畫面確認，告知提取金鑰。

3 用戶端公鑰管理(銀行或 PKI 維運單位可以自行決議)

涉及無法公開之公鑰演算程序者，例如：AES 加密協定，若對公鑰進行安全防護，協作設計建議同上開(一)之 2、(二)與(三)之解決方案。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.3. 敏感性資料儲存

行動應用 App 基本資安檢測基準：

4.1.2.3.7. 檢查行動應用程式之程式碼或其他封裝之檔案內容，是否未檢出密碼、身分驗證資訊或對稱式加解密演算法之金鑰。

各國開發要點參考： ENISA SSDG]2.5,2.10,NIST 800 SP-163 3.1.4.

參考來源：N/A

CPF-13:當提供使用者選擇儲存使用者 ID，來加速日後身分驗證便利性，應實作加密方法以保護真實 ID，而顯示時可使用選項有遮罩(Mask)或、代碼化(Tokenization)及雜湊(Hash)等。

說明：

依 Android 訓練教材，使用 Android ID 建議時遵循以下原則：

#1：避免使用硬體 ID

硬體 ID，例如：SSAID(Android 的 ID)，和 IMEI 在大多數使用情況即使不使用它們，也能達成需要的功能。

#2：僅使用廣告 ID 作為使用者分析或廣告使用

在使用廣告 ID(查詢時間：2016/10/1)，請尊重使用者限制廣告跟蹤(查詢時間：2016/10/1)設定，確保此 ID 不能連接到個人身分資訊(PII)且避免將使用者重設的廣告 ID 原有的廣告 ID 做連結。

#3：為了防詐欺支付和詐欺電信在大多數的情形，儘可能使用實例(Instance)ID 或私下儲存 GUID 。

對於絕大多數的非廣告使用案例中，實例 ID 或 GUID 應該是可以滿足應用。

#4：開發人員依使用情況使用合適的 API，以盡量減少隱私風險 。

當需要對高價值的數位版權內容保護時，可使用 DRM API.(查詢時間：2016/10/1)

而要防止 App 被濫用時可使用 SafetyNet API(查詢時間：2016/10/1)

。該 API 提供正確及簡單的方法來確定設備是否為原廠設定或被惡意入侵，從而造成資安風險 。

詳細使用情境與建議詳如 Best Practices for Unique Identifiers(查詢時間：2016/10/1) 在 iOS 上，當使用者啟用了"保存此使用者 ID"功能，使用者名是 CredentialsManager 對象中暫存檔。在運行時，使用者名被上傳到記憶體中的任何類型的身分驗證之前發生，讓潛在惡意程序攔截的使用者名。為了保護使用者資料，不要 LOG 使用者名稱。

在 Android 上，請儘可能不要使用 Android ID 或 Device ID 等這些硬體 ID，而改以其他的虛擬 ID 來做為辨識設備或使用者的 ID。

建議改用 HASH 與 UUID 進行手機端驗證，並可避免換手機後，帳號仍可驗證的問題。

建議不儲存 UserName，改以 Hash+UUID 進行 Client profile 之驗證，也防止換環境執行之情況.

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

Android：Java

以下範例說明如何產生一個 UUID，以供 APP 認定此設備為原有設備。

```
//在第一次安裝完後，產生一組 UUID
String uniqueId = java.util.UUID.randomUUID().toString();
String hashId = encrypt(uniqueId);
```

```
//加密完成後，送到主機儲存，以供綁定此設備
sendToServerToSave(hashId);
//另儲存於設備上，以供後續比對
saveHashedId(hashId);
……..
……..
//後續如需檢查時將從主機取得已經 hash 過的 ID， 和本機儲存的 ID 進行比較，確認是否為同一台設備
if(! getHasedIdFromServer().equals(getSavedHashedId())){
        //當不相同的時候，採取相對應的作法，例如：要求使用者，登入遠端主機，認証此設備
}
   public static String encrypt(String s){
        MessageDigest sha = null;
        try{
            sha = MessageDigest.getInstance("SHA-256");
            sha.update(s.getBytes());
        }catch(Exception e){
            e.printStackTrace();
            return "";
        }
        return byte2hex(sha.digest());
    }
    private static String byte2hex(byte[] bytes){
        String hs="";
        String stmp="";
        for (byte myByte: bytes){
            stmp=(java.lang.Integer.toHexString(myByte & 0XFF));
            if (stmp.length()==1) hs=hs+"0"+stmp;
            else hs=hs+stmp;
        }
        return hs.toUpperCase();
    }
```

使用 Google Play 服務提供的 SafetyNet API，來檢查設備是否有被 Root 過，或被篡改。

```
• byte[] nonce = getRequestNonce(); // Should be at least 16 bytes in length.
  SafetyNet.SafetyNetApi.attest(mGoogleApiClient, nonce)
          .setResultCallback(new ResultCallback<SafetyNetApi.AttestationResult>()
  {
```

```
        @Override
        public void onResult(SafetyNetApi.AttestationResult result) {
            Status status = result.getStatus();
            if (status.isSuccess()) {
                // 通訊成功
                String data = result.getJwsResult();

•               // 對 data 進行相關解碼，取得"ctsProfileMatch": true 則表示設備正常，反之
  則表示設備不安全

            } else {
                // An error occurred while communicating with the service
            }
        }
    });
```

iOS：Swift

```
通常會使用第三方帳號認證，茲提供OAuth2 範例，以取得第三方的Token，當作認證
ID
OAuth 2 應用範例
1.註冊並宣告服務
let googleConfig = GoogleConfig(
clientId: "YOUR_GOOGLE_CLIENT_ID",                          // [1] Define a Google
configuration
scopes:["https://www.googleapis.com/auth/drive"])             // [2] Specify scope

let gdModule = AccountManager.addGoogleAccount(googleConfig)     // [3] Add it to
AccountManager
self.http.authzModule = gdModule                              // [4] Inject the
AuthzModule
                                                          // into the HTTP Layer
object

let multipartData = MultiPartData(data: self.snapshot(),        // [5] Define multi-
part
        name: "image",
```

```
          filename: "incognito_photo",
          mimeType: "image/jpg")
let multipartArray =  ["file": multipartData]

self.http.POST("https://www.googleapis.com/upload/drive/v2/files",   // [6] Upload
image
              parameters: multipartArray,
       completionHandler: {(response, error) in
  if (error != nil) {
    self.presentAlert("Error", message: error!.localizedDescription)
  } else {
    self.presentAlert("Success", message: "Successfully uploaded!")
  }
})
```

註冊 App 應用連結

```
<key>CFBundleURLTypes</key>
<array>
    <dict>
        <key>CFBundleURLSchemes</key>
        <array>
            <string>com.raywenderlich.Incognito</string>
        </array>
    </dict>
</array>
```

3.接收 App 應用連結

```
func application(application: UIApplication,
  openURL url: NSURL,
  sourceApplication: String?,
  annotation: AnyObject?) -> Bool {
    let notification = NSNotification(name: AGAppLaunchedWithURLNotification,
      object:nil,
      userInfo:[UIApplicationLaunchOptionsURLKey:url])
    NSNotificationCenter.defaultCenter().postNotification(notification)
    return true
}
```

行動應用 App 基本資安規範：4.1.2.3.敏感性資料儲存

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 2.1,2.2

參考來源：

Android： Best Practices for Unique Identifiers(查詢時間：2016/10/1)

iOS：

OAuth 2.0 with Swift Tutorial (查詢時間：2016/10/1)

http://resources.infosecinstitute.com/iOS-application-security-part-20-local-data-storage(查詢時間：2016/10/1)

CWE/OWASP：

M2 - Insecure Data Storage(查詢時間：2016/10/1)

M4 - Unintended Data Leakage(查詢時間：2016/10/1)

CWE 312, 313, 522(查詢時間：2016/10/1)

CPF-14:使用設定或編寫程式停用對敏感資料鍵盤輸入資料的自動更正(auto-correct)功能。

說明：

因主機端有文字輸入的歷史資料功能，為了保護資料，機敏資料的輸入，應關閉這些機制，以免資料洩漏。

Android 的使用者辭典，在使用者可以保存為未來輸入文字的自動校正。此使用者辭典可用於任何應用，而無需特殊權限。為了提高安全性，可以考慮實施自行定義鍵盤(和潛在的 PIN 碼輸入)，它可以禁用快取，並提供針對惡意軟體攻擊提供額外的保護。

iOS 的記錄個別使用者輸入習慣，為了提供諸如客製自動校正和形式完成鍵入，但敏感的資料也可被儲存。幾乎每一個非數字在它被輸入的順序鍵盤暫存檔中;高速暫存檔的內容超出了應用程式的管理權限，並因此資料不能從應用程式暫存檔中移除。

禁用自動更正功能，對任何敏感資訊，而不僅是密碼字段。由於鍵盤暫存檔中的敏感資訊，是可以被恢復的。對於的 UITextField，窺視 autocorrectionType 屬性設置為 UITextAutocorrec-tionTypeNo 禁用暫存檔。這些設置可能會隨時間而改變的 SDK 更新，以便確保其充分的研究。添加企業政策來定期重置鍵盤辭典。這可以由最終使用者通過簡單地去設置應用程式來完成，通用>還原>還原鍵盤辭典。

Android 的使用者辭典，在使用者輸入文字可以保存作為未來的自動校正使用。此使用者辭典可為任何應用程式存取，而無需特殊權限。為了提高安全性，可以考慮實施自定義鍵盤(和潛在的 PIN 碼輸入)，它可以禁用快取，或停用此功能提供針對惡意軟體提供額外的保護。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

Android：Java

此範例分別說明如何在 XML 檔或是 Java 程式碼內設定停用自動校正

In XML 使用 android:inputType="textNoSuggestions" 停用此功能

```
<EditText android:id="@+id/edtInput"
        android:layout_width="0dip"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:inputType="textNoSuggestions"
        android:maxLines="4"
        android:maxLength="2000"/>
```

In Java Code 使用 InputType.TYPE_TEXT_FLAG_NO_ SUGGESTIONS 停用此功能

```
edtInput.setInputType(InputType.TYPE_TEXT_FLAG_NO_SUGGESTIONS);
```

iOS：Objective-C

- 建議關閉文字輸入框的自動校正功能，以免資料洩漏
- myInput.autoCorrectionType = UITextAutocorrectionTypeNo;
- 建議關閉文字輸入框的拼字校正功能，以免資料洩漏
- myInput.spellCheckingType = UITextSpellCheckingTypeNo;
- 並可由手機設定中心，清除輸入的紀錄

General > Reset > Reset Keyboard Dictionary.

行動應用 App 基本資安規範：4.1.2.3.敏感性資料儲存

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：

Android：

https://developer.android.com/reference/android/text/InputType.html (查詢時間：2016/10/1)

CWE/OWASP

M4 - Unintended Data Leakage(查詢時間：2016/10/1)

CWE 200(查詢時間：2016/10/1)

CPF-15:使用設定或編寫程式停用敏感資料區域的複製及貼上功能。

說明：

iOS 和 Android 的支援複製/貼上(copy/paste)功能。剪貼板不管該資料的來源最初是否加密，敏感資料可被儲存、恢復，或者可以從明文剪貼板被修改。如果當使用者從其他應用程式存取剪貼板，拷貝當時是明文，貼上也會是明文。

在適當情況下，禁用複製/貼上處理敏感資料區域粘貼。消除複製選項可以避免資料洩露。在 Android 剪貼板可以由任何應用程式進行存取，因此建議適當設定內容提供商用於傳輸複雜的敏感資料。在 iOS 上考慮使用者是否需要複製/貼上應用程式或系統範圍內的資料，並選擇適當的類型template。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

Android: Java

根據使用情境不同，以下提供二種方式來防止複製/貼上的行為

1.停止元件的剪下/複製功能

```
import android.app.Activity;
import android.os.Bundle;
import android.support.v4.app.NavUtils; import android.view.ActionMode;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.EditText;
public class UncopyableActivity extends Activity { private EditText copyableEdit;
private EditText uncopyableEdit;
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState); setContentView(R.layout.uncopyable);
copyableEdit = (EditText) findViewById(R.id.copyable_edit); uncopyableEdit =
(EditText) findViewById(R.id.uncopyable_edit);
// By setCustomSelectionActionMODECallback method,
// Possible to customize menu of character string selection.
uncopyableEdit.setCustomSelectionActionModeCallback(actionModeCallback);
}
private ActionMode.Callback actionModeCallback = new ActionMode.Callback() { public
boolean onPrepareActionMode(ActionMode mode, Menu menu) {
return false; }
public void onDestroyActionMode(ActionMode mode) { }
public boolean onCreateActionMode(ActionMode mode, Menu menu) {
// *** 從功能選單刪除 android.R.id.copy
```

```
MenuItem itemCopy = menu.findItem(android.R.id.copy);
if (itemCopy != null) {
menu.removeItem(android.R.id.copy); }
// *** 從功能選單刪除 android.R.id.cut

MenuItem itemCut = menu.findItem(android.R.id.cut); if (itemCut != null) {
menu.removeItem(android.R.id.cut); }
return true; }
public boolean onActionItemClicked(ActionMode mode, MenuItem item) { return false;
} };
@Override
public boolean onCreateOptionsMenu(Menu menu) {
getMenuInflater().inflate(R.menu.uncopyable, menu);
return true; }
@Override
public boolean onOptionsItemSelected(MenuItem item) {
switch (item.getItemId()) { case android.R.id.home:
NavUtils.navigateUpFromSameTask(this); return true;
}
return super.onOptionsItemSelected(item); }
}
```

2.停止長按彈出視窗

將android:longClickable 設為false

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
android:layout_height="match_parent"

android:orientation="vertical">

<TextView

android:layout_width="match_parent" android:layout_height="wrap_content"
android:text="@string/unlongclickable_description" />

<!-- EditText to prohibit copy/cut EditText -->

<!-- *** 將 android:longClickable設為False 來禁止. --> <EditText

android:layout_width="match_parent"
```

```
android:layout_height="wrap_content" android:longClickable="false"
android:hint="@string/unlongclickable_hint" />

</LinearLayout>
```

iOS：Objective-C

```
• 監聽 TextField/TextView 事件
• @interface ViewControllerPaste : UIViewController <UITextFieldDelegate,
  UITextViewDelegate>
• 當敏感輸入元件有事件(Copy 等)時，不允許 Copy/Paste
• - (BOOL)canPerformAction:(SEL)action withSender:(id)sender {
•    //identy those who not allow copy/paste
•    [myInput resignFirstResponder];
•    return NO;
• }
```

行動應用 App 基本資安規範：4.1.2.3.敏感性資料儲存

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：JSSEC AASDSCG Risk of Information Leakage from Clipboard

參考來源：N/A

CPF-16:當行動應用 App 使用行動智慧裝置攝影鏡頭拍攝敏感性資料，如身分證、財務文件，應僅暫存於記憶體不儲存於本機檔案系統，完成傳輸後即由記憶體清除。

說明：

某些金融創新服務，允許使用相機拍攝交易憑證，傳送給遠端金融機構辦理交易，如支票存款應用程式，允許使用者使用手機的相機功能拍攝支票，並把它傳送給金融機構，以達成將支票存入帳戶的交易行為。許多此類型應用程式就算刪除圖片後，仍留下部分或全部圖片於手機的 NAND 記憶體。

注意：

建議採用不同的方法來避免在非暫存的儲存設備中留下支票圖片。在 Android 上可以建立一個 SurfaceView，並讓他呈現一個相機預覽(preview)，或鏡頭所看到的即時預覽。然後按鈕按下時，此預覽畫面會轉成一個像素陣列的資料回傳。

接下來就可以再轉換成 Bitmap,再壓縮成 JPG 圖檔，採用 base64 編碼成文字後上傳給雲端主機。這個流程中圖像僅在暫時記憶的 RAM 中，以避免圖像的可能被暫存檔到非逸失性儲存設備(內置或外存的 flash 記憶體)。

在 Android Camera 類別內，takePicture()可用於指定一 Callback.所以當使用 Camera.PictureCallback Interface 產出 JPEG 時將會呼叫此一 Callback 方法。

下列這個方法也是很重要的：

public void onPictureTaken(byte[] bytes, Camera camera)。

使用這個方式可以使用位元陣列來儲存相片內容，而此陣列只會存在於 RAM 中。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.3.敏感性資料儲存

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：N/A

CPF-17:行動應用 App 有提供標準的設定參數檔函數，但由於駭客取得安裝檔之後，非常容易就可以修改這些標準的參數檔，於是為了加強安全層級，應用程式的設定參數，建議放在安全的地方，例如：編譯至程式碼中，或者進行加密。

說明：

iOS 開發常儲存應用程式的設定在可以在某些情況下會受到影響的 plist 文件。同樣地 Android 的開發商往往保存在共享的喜好 XML 文件或 SQLite 資料庫設定，預設情況下不會加密，並可以讀取或甚至 root 權限進行修改。

在可行的情況下儘可能編譯設定到程式碼。有一點好處是於通過配置在 iOS plist 文件的應用程式，因為變更必須捆綁，且無論如何均需部署為新的應用程式。相反地攻擊者需要包括更多的時間和技能，以修改應用程式程式碼中的配置。不要存放在檔案目錄或其他文件中包含任何密鑰的設定，除非先進行加密。理想的情況下，使用由使用者提供的密碼的主密鑰，去加密所有的設定文件，或由遠端提供的一個密鑰。

安全開發生命週期：開發實作階段

不安全程式碼範例：

iOS：Objective-C

Application Setting Read Example

將設定置於 plist 設定檔，容易被破解後修改

```
    //取得 plist 檔案路徑
    NSString *filePath = [[NSBundle mainBundle] pathForResource:@"secrets" ofType:@"plist"];
    NSFileManager *fileManager = [NSFileManager defaultManager];
    [self AppendLog:[NSString stringWithFormat:@"filePath=%@", filePath]];
    //判斷 plist 檔案存在才讀取
    if ([fileManager fileExistsAtPath: filePath]) {
        NSMutableDictionary *data = [[NSMutableDictionary alloc] initWithContentsOfFile:filePath];
        NSString *hash_password = [data objectForKey:@"hash_password"]; //從 data 中取值
        NSString *setting_lifes = [data objectForKey:@"setting_lifes"]; //從 data 中取值
    } else{
        [self AppendLog:@"file not exists"];
    }
```

安全程式碼範例：

Android：Java

此範例說明如何將加密儲存於 property file 的密碼來和使用者輸入的密碼做安全的比對

```
public class MainActivity extends ActionBarActivity {
```

```
    String flag = "com.ereach.helloworld";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);



String encryptPassword = loadPassword();
//因此加密方式為不可逆，如需要比較密碼時，請將使用輸入的密碼加密後比對
// if(encrypt(userInputPassword).equals(encryptPassword))
    }
public static String encrypt(String s){
        MessageDigest sha = null;
        try{
            sha = MessageDigest.getInstance("SHA-256");
            sha.update(s.getBytes());
        }catch(Exception e){
            e.printStackTrace();
            return "";
        }
        return byte2hex(sha.digest());
    }
    private static String byte2hex(byte[] bytes){
        String hs="";
        String stmp="";
        for (byte myByte: bytes){
            stmp=(java.lang.Integer.toHexString(myByte & 0XFF));
            if (stmp.length()==1) hs=hs+"0"+stmp;
            else hs=hs+stmp;
        }
        return hs.toUpperCase();
    }
 //讀取資料的方法
private String loadPassword(){
//建構 properties 物件
Properties properties = new Properties();
    try{
        FileInputStream stream = this.openFileInput("music.cfg");
```

```
        //讀取兩個檔的內容
        properties.load(stream);
        properties.load(stream1);


    }catch(FileNotFoundException e)
    {
    return;
    }catch(IOException e){
    return;
    }
    return String.valueOf(properties.get("password"));
}
}
```

iOS：Objective-C

protect application parameter : example code

- *將設定或參數置於程式碼，或進行隱藏*

```
//Stored as String in program, still not so safe to cracker
NSString *APIKey = @"abcdef123456";
// Stored as binary array to prevent strings attack
 // Stored using SHA(class name) XOR secret
 unsigned char obfuscatedSecretKey[] = { 0x3e, 0xcc, 0x9d, 0xca, 0x2f, 0x8a, 0xf2,
0xc1, 0x57, 0x01, 0xc2, 0x2e, 0x44, 0xea, 0x0d, 0xf5, 0x60, 0x09, 0x99, 0xe7, 0xb1,
0x13, 0x2a, 0x6f, 0x2c, 0xa2, 0xfe, 0x12, 0xbb, 0x58, 0x90, 0x85 };
```

行動應用 App 基本資安規範：4.1.2.3.敏感性資料儲存

行動應用 App 基本資安檢測基準： N/A

各國開發要點參考：N/A

參考來源：CWE/OWASP

M2 - Insecure Data Storage, M4 - Unintended Data Leakage(查詢時間：2016/10/1)

CWE 312, 313(查詢時間：2016/10/1)

CPF-18:使用行動應用 App Cookies 的安全設定。

說明：

網站常使用 Cookie 紀錄當作溝通橋樑，若 Cookie 未標記安全參數，可能會通過不安全的傳輸管道進行傳送。換句話說，它可以允許於 HTTP 連接中使用。建議在 Cookie 加上 Secure 參數，使得 Cookie 只能應用在 HTTPS 之上，已達到保護資料的目的。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

Android：Java

因主機端已對 Cookie 進行安全設定，故設備端需配合偵測 Cookie 是否為 httpOnly/requireSSL，來進一步確認此 Cookie 是否同時具有此設定，注意：從 Android API22 開始 HttpClient 已經棄用，新的 HttpCookie 無法檢查 isSecure

```
CookieStore cookieStore = java.net.CookieManager.getCookieStore();
       if (cookieStore != null) {
           List<HttpCookie> cookies = cookieJar.getCookies();
           for (HttpCookie httpCookie : cookies) {

               Log.i("HttpOnly", httpCookie.isHttpOnly());
           }
       }
```

iOS：Objective-C

偵測 Cookie 是否為 httpOnly/requireSSL

```
監聽NSURLConnect事件
- (void)connection:(NSURLConnection *)connection didReceiveResponse:(NSURLResponse
*)response
{
    NSLog(@"--------------didReceiveResponse dump NSHTTPCookieStorage------------
\n");
    for (NSHTTPCookie *cookie in [[NSHTTPCookieStorage sharedHTTPCookieStorage]
cookies])
    {
        NSLog(@"name: '%@'\n",   [cookie name]);
        NSLog(@"value: '%@'\n",  [cookie value]);
        NSLog(@"domain: '%@'\n", [cookie domain]);
        NSLog(@"path: '%@'\n",   [cookie path]);
```

```
        NSLog(@"HttpOnly,Secure: '%d,%d'\n",   [cookie isHTTPOnly], [cookie
isSecure]);
    }
}
```

主機端：Server

輸出網頁內容時，設定 Cookie Http-Only 與 Secure 旗標，以保持安全。

```
<httpCookies httpOnlyCookies="true" …>
<httpCookies requireSSL="true" />
```

行動應用 App 基本資安規範：4.1.2.3.敏感性資料儲存

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：CWE/OWASP：

M9 - Improper Session Handling(查詢時間：2016/10/1)

CWE 614(查詢時間：2016/10/1)

3.1.7. 敏感性資料傳輸(G)

CPG-01:行動應用 App 傳輸敏感性資料應規劃並實作傳輸全程全時使用 TLS 1.1 或以上加密，維護敏感性資料機密性及完整性。

說明：

過去網頁瀏覽考慮在網際網路或任何形式無線傳輸傳輸敏感性資料，都會以 SSL/TLS 加密作為優先考量，一般使用者也分不清楚 SSL/TLS 的差異，二者都是以相同的方式確保安全連接 ，它們使用安全憑證以進行認證，以及使用公鑰/私鑰加密，然而雙方仍存在著建置上的差異性。需注意的新的 Web 應用協定 HTTP/2 和 SPDY 並不支援 SSL。它們只支援 TLS。

Google 所有公共服務的加密都是使用 TLS。如果要整合 Google 的一些功能，就應注意 Google 使用的是 TLS 而非 SSL。因為這有著實質上的差異在支付卡產業資料安全標準(PCI DSS)規範下，SSL 在 2016/6/30 之後不得繼續使用。那個時間點之後，將明確要求採用 TLS。

針對資料完整性部分，也可以增加押碼或雜湊運算等方式，提供進一步完整性驗證。

iOS 可同時多項支援傳輸層安全性協定(TLS v1.0、TLS v1.1、TLS v1.2)和 DTLS。高階 API(如 CFNetwork)讓開發人員可以輕鬆在其 App 中採用 TLS，而低階 API(SecureTransport)則提供精細的控制。依照預設 CFNetwork 不允許 SSLv3，而使用 WebKit 的 App(如 Safari)也被禁止進行 SSLv3 連線。

「App 傳輸安全性」提供預設連線的需求，以便 App 在使用 NSURLConnection、CFURL 或 NSURLSession API 時，遵循安全連線的最佳做法。

伺服器必須支援 TLS 1.2 的最低限度(Forward Secrecy)，且憑證必須有效並使用 SHA-256 加以簽署(最好使用 2048 位元 RSA 密鑰或 256 位元橢圓曲線密鑰的最低限度)。

Apple 將在 2017 年 1 月強制所有 iOS App 開始使用 ATS(App Transport Security,支援 TLS v1.2)，參考 iOS-04。

Android 支援標準 TLS 實作，在官方開發人員網站有完整的概念及實作介紹，並包含實例及常見問題。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.4.敏感性資料傳輸

行動應用 App 基本資安檢測基準

4.1.2.4.1.(1) 檢查行動應用程式是否採用 TLS 1.1(含)以上版本加密協定傳輸敏感性資料。

各國開發要點參考：JSSEC AASDSCG .5.4

參考來源：

Android：[Security with HTTPS and SSL(查詢時間：2016/10/1)]

iOS：

iOS Security Guide P.28,May 2016(查詢時間：2016/10/1)

用 HTTPS 就萬無一失？其實你可能用的是 TLS 而不是 SSL(查詢時間：2016/10/1)

Transport Layer Security,Wiki(查詢時間：2016/10/1)

CPG-02:行動應用 App 應使用作業系統平台或可信任來源提供最新版 TLS API 及安全實作規範，實作全時全程安全傳輸管道。

說明：

現今網路層的攻擊有能力可以解密電信服務提供商網路加密機制，並且沒有保證在連接 Wi-Fi 網路時一定可以適當地加密。

不論是否傳輸敏感資料與否，強制使用一個終端到終端(end to end)的加密安全通道。這包括通過使用者憑證或其他驗證因子。實作全時全程機密性和完整性的傳輸保護。

iOS 支援業界標準的 Wi-Fi 通訊協定，包括「WPA2 企業級」，可針對無線企業網路提供連線認證服務。「WPA2 企業級」使用 128 位元 AES 加密，可為使用者提供最高等級的安全保障：在透過 Wi-Fi 網路連線傳送和接收通訊時，確保使用者的資料始終受到機密性和完整性保護。有了 802.1X 的支援，iOS 裝置可被整合到各種 RADIUS 認證環境中。iPhone 和 iPad 上支援的 802.1X 無線認證方式包括 EAP-TLS、EAP-TTLS、EAP-FAST、EAP-SIM、PEAPv0、PEAPv1 及 LEAP。

安全開發生命週期：開發實作階段

不安全程式碼範例：

Android：Java

假設呼叫 getInputStream()時它拋出一個異常訊息，而不是收到的 getInputStream()的內容：

```
• javax.net.ssl.SSLHandshakeException:
  java.security.cert.CertPathValidatorException: Trust anchor for certification
  path not found.

•         at
  org.apache.harmony.xnet.provider.jsse.OpenSSLSocketImpl.startHandshake(OpenSSLS
  ocketImpl.java:374)

•         at
  libcore.net.http.HttpConnection.setupSecureSocket(HttpConnection.java:209)

•         at
  libcore.net.http.HttpsURLConnectionImpl$HttpsEngine.makeSslConnection(HttpsURLC
  onnectionImpl.java:478)

•         at libcore.net.http.HttpsURLConnectionImpl$HttpsEngine.connect
            (HttpsURLConnectionImpl.java:433)

•         at libcore.net.http.HttpEngine.sendSocketRequest(HttpEngine.java:290)
```

```
•       at libcore.net.http.HttpEngine.sendRequest(HttpEngine.java:240)
•       at libcore.net.http.HttpURLConnectionImpl.getResponse(HttpURLConnectionImpl.java:282)
•       at libcore.net.http.HttpURLConnectionImpl.getInputStream(HttpURLConnectionImpl.java:177)
•       at libcore.net.http.HttpsURLConnectionImpl.getInputStream(HttpsURLConnectionImpl.java:271)
```

發生這種情況的原因很多，其中包括：發行 Server 憑證的 CA 不是由具有公信力 CA 簽名，但自簽名伺服器配置缺少中間 CA 無法確認可以是安全的，最好的方式還是綁定可信任的憑證，憑證綁定實作，請參考 CPM-05。

安全程式碼範例：

Android: Java

假設你有由信任 CA 頒發的證書的 Web 伺服器，你可以用 URLConnection 來進行安全要求：

```
URL url = new URL("https://wikipedia.org");
URLConnection urlConnection = url.openConnection();
InputStream in = urlConnection.getInputStream();
copyInputStreamToOutputStream(in, System.out);
```

行動應用 App 基本資安規範：4.1.2.4.敏感性資料傳輸

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 3.1,ENISA SSDG 3.2

參考來源：Android：Security with HTTPS and SSL(查詢時間：2016/10/1)

CPG-03:行動應用 App 實作 CPG-02 TLS 加密應使用或於加密模組驗證計畫(CRYPTOGRAPHIC MODULE VALIDATION PROGRAM,CMVP)適用性清單中加密模組或知名加密演算法並採用足夠長度加密金鑰。

(a)採用金鑰有效長度為 2048 位元(含)以上之 RSA 加密演算法，或採用金鑰有效長度為 224 位元(含)以上之橢圓曲線加密演算法(Elliptic Curve Cryptography)。

(b)採用金鑰有效長度為 128 位元、192 位元或 256 位元之進階加密標準(AES)，或採用金鑰有效長度為 112 位元或 168 位元之三重資料加密演算法(Triple DES)。

說明：

Android 使用 Java API(Oracle 維護)實作，可選擇 AES 演算法，128 位元、192 位元或 256 位元的金鑰長度，支援 TLS V1.0,V1.1,V1.2。詳細加解密參數請參考 Java Cryptography Architecture Standard Algorithm Name Documentation for JDK 8 (查詢時間：2016/10/1)。

iOS 參考 CPG-01 及 iOS-003

以上 Android 及 iOS 加密模組均符合 CMVP

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.4.敏感性資料傳輸

行動應用 App 基本資安檢測基準：

4.1.2.4.1.(2) 檢查行動應用程式是否採用金鑰有效長度為 2048 位元(含)以上之 RSA 加密演算法，或採用金鑰有效長度為 224 位元(含)以上之橢圓曲線加密演算法(Elliptic Curve Cryptography)。

4.1.2.4.1.(3) 檢查行動應用程式是否採用金鑰有效長度為 128 位元、192 位元或 256 位元之進階加密標準(AES)，或採用金鑰有效長度為 112 位元或 168 位元之三重資料加密演算法(Triple DES)。

各國開發要點參考：CSA MAST 4.2.6, ENISA SSDG 3.3, NIST 800 SP-163 3.1.4, JSSEC AASDSCG .5.4

參考來源：

Android：Java Cryptography Architecture Standard Algorithm Name Documentation for JDK 8(查詢時間：2016/10/1)

3.1.8. 敏感性資料分享(H)

CPH-01:將敏感性資料分享給不同行動應用 App 應實作使用者同意或拒絕選項，提示使用者選擇時機可於(1)安裝時(2)當敏感資料被儲存或傳送前(3)預設設定為關閉同意，需使用者自行開啟。

說明：

參考 CPA-01 將行動應用 App 需要分享敏感性資料類別於 Google Play 及 App Stroe 及行動應用程式的隱私權政策中聲明。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.5.敏感性資料分享

行動應用 App 基本資安檢測基準：

4.1.2.5.1.(1)檢查行動裝置內之不同行動應用程式間，分享敏感性資料前，是否於行動應用程式內或可信任之應用程式商店聲明。

4.1.2.5.1.(2) 檢查行動裝置內之不同行動應用程式間，分享敏感性資料前，是否於行動應用程式內或可信任之應用程式商店取得使用者同意。

各國開發要點參考：ENISA SSDG 7.2 , JSSEC AASDSCG .5.4

參考來源：N/A

CPH-02:當使用者已於行動應用 App 選取拒絕分享敏感性資料給其他行動應用 App 選項，不應實作於背景中運作將敏感性資料以任何形式連線機制傳送給其他應用程式或不明目的地。

說明：

在社群類型行動應用 App，如 facebook、Twitter、line 等分享功能即常用到的隱私設定，通常是會預設為分享，如要製作類似行動應用 App，建議仍是要參考 CPD-06 做法，預設為關閉自動分享及傳輸至其他應用程式，只有在需要分享時，於使用者介面實作彈出選單供使用者選擇，在使用者沒有明確意願前，不得於 App 背景實作將敏感性資料以任何形式連線機制傳送給其他應用程式或不明目的地。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.5.敏感性資料分享

行動應用 App 基本資安檢測基準：

4.1.2.5.1.(3)檢查在未聲明或未取得使用者同意敏感性資料分享的情況下，行動應用程式是否未分享敏感性資料予行動裝置內之不同行動應用程式。

4.1.2.5.2.(1)檢查行動應用程式是否提供使用者拒絕分享敏感性資料之選項。

4.1.2.5.2.(2)檢查在使用者拒絕敏感性資料分享的情況下，行動應用程式是否未分享敏感性資料

4.1.2.5.3.檢查分享敏感性資料之行動應用程式，是否限定特定行動應用程式可存取敏感性資料。

各國開發要點參考：NIST 800 SP-163 3.1.2,3.14,3.1.5.g ,ENISA SSDG 3.7 ,7.4

參考來源：N/A

CPH-03:在分享敏感性資料前，要注意驗證接收來自或傳送至的目的地是否為信任可靠。

說明：

在數位化行銷盛行年代，行動應用 App 通常不會只限於孤島式自行運作，常見如新聞的 App，在使用者閱讀後，會讓使用者可以分享至像 facebook、line 等社群軟體，在不同 App 間傳送資料是常態。

如果需要傳送或接收敏感性資料前，所以開發人員必須謹慎驗證接收來源或傳送目的地是否安全可靠。

Android 的在 App 間的通訊(Interprocess Communication, IPC)程序可以使用傳統 Unix 的機制，如 filesystem, local sockets,或 signals 的權限管理，Android 也提供 Binder(實作請參考 https://developer.android.com/reference/android/os/Binder.html(查詢時間：2016/10/1)，Intents (參考 Android-01)、Services(Android-06)、ContentProviders(Andrid-008)，在實作上需要保護不會被惡意攻擊者注入或是篡改傳送至不明目的地。

在 iOS 因為沙箱隔離，App 間的資料分享是以 URL Schemes 來實作，也是需要注意其他 App 或遠端通訊的另一端是否仍保持安全可信任。以下為幾種呼叫方式的風險與對策：

Mach Messaging:如使用 Mach Messaging，避免提供 App 本身程序使用的通訊埠，因為訊息有可能被修改為執行任意程式碼，如果需要應該為 client 端另建一個專用通訊埠。

Remote procedure calls (RPC) and Distributed Objects:RPC 通訊是起源甚早且簡單的 Client-Server 架構的通訊協定，由 Client 發出 request 再由 Server 端回應，通訊之間沒有任何保護下，也是相當容易被修改為可執行惡意任意碼，除非是在內部網路且可信任的端點外，應該避免使用 RPC。

Shared Memory:如果想要與其他 App 共享記憶體，可能會因為實作不當造成不正常運作，或遭注入惡意程式碼，另外可能造成 race condition 攻擊，記憶體對映文件亦有可能被置換為惡意，所以必須保護。

最後，命名為共享儲存器區域和儲存器映射文件可以通過運行在使用者的任何其他方法來存取。基於這個原因，它是不使用安全的非匿名共享儲存器程序之間發送高度機密的資訊。而是，分配之前創建子程序，需要共享區域的共享儲存區域，然後通過 IPC_PRIVATE 作為鍵 shmget，以確保共享儲存器 ID 不是容易猜測。

Signals:程序間可以用信號(Signals)來傳遞工作要求及回應要求，攻擊者有可能透過緩衝區溢位來注入惡意執行命令。出於這個原因，信號處理工作量，應該儘可能最小化，並應在應用程式的主程序循環中的一個已知位置執行批次的工作。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.5.敏感性資料分享

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 6.3

參考來源：

Android：Interprocess Communication（查詢時間：2016/10/1）

iOS：Interprocess Communication and Networking（查詢時間：2016/10/1）

3.1.9. 敏感性資料刪除(I)

CPI-01:行動應用 App 儲存敏感性資料，應預先針對各國個人資料保護法或隱私要求及實務需求，定義資料於行動智慧裝置合理最長保存期限，預設 App 自使用者行動智慧裝置刪除或反安裝同時，針已儲存持久性敏感資料屆期可詢問是否需要刪除或是進行永久性加密。

說明：

資料刪除或銷毀是使用者敏感性資料離開行動應用App 生命週期的終點站，行動應用App 的內、外部儲存空間有限，非持久性資料，如地理位置、日誌檔等應「設定留存時間；持久性敏感性資料，如個人基本資料、聯絡人資料及付費資訊等，則需考量各國個人資料保護法或隱私要求及實務需求，定義資料於行動智慧裝置合理最長保存期限，預設為 App 自使用者行動智慧裝置刪除或反安裝同時，並將敏感性資料保留期限，納入行動應用程式商店及行動應用 App 內隱私權政策聲明中，讓使用者了解。

安全開發生命週期：需求階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.6.敏感性資料刪除

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：N/A

CPI-02:行動應用 App 本地端及遠端伺服器暫存檔、暫存及儲存於記憶體或內外部儲存媒體敏感性資料，應依 CPI-01 定義敏感性資料最長保留期限設計及實作提供使用者選擇清除及永久性刪除功能。

說明：

在系統設計階段，系統設計人員應依 CPI-01 定義各類敏感性資料保留時間，區分本地端及遠端伺服器暫存檔、暫存及儲存於記憶體或內外部儲存媒體敏感性資料，屆期或當使用者刪除行動應用程式。設計原則建議暫存於記憶體或暫存性質非持久敏感性資料，屆期應自動清除，不用提示使用者。

儲存於本地端 App 儲存區域，應於屆期或使用者移除 App 同時，應設計提示使用者清除或是同意繼續保留機制，除清除 App 本地端外並需要通知後端伺服器資料庫管理員進行清除(CPI-06)，清除應達不可復原的水準。

安全開發生命週期：設計階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.6.敏感性資料刪除

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG1.6, ENISA SSDG1.8

參考來源：N/A

CPI-03:於行動智慧裝置 NAND flash 記憶體，刪除敏感性資料除呼叫檔案刪除 API 不保證資料可完全被清除，可以加密提高資料被復原難度作為替代方案。

說明：

在 Android 上呼叫 file.delete()並不會真正把資料刪除，只要原資料儲存區，尚未被其他檔案覆蓋前，仍然是可以被復原，傳統的以大量資料覆蓋行動智慧裝置 NAND flash 記憶體，將迫使 NAND flash 記憶體抹除所有未配置空間，將會造成行動智慧裝置運行變得緩慢且耗電，將資料加密且做好安全密鑰管理是比較實務的做替代方案。Android 在 Lollipop(5.0)之後的版本，預設為全機加密，可以在使用遠端抹除將資料清除，回復出廠預設值。

在 iOS 上檔案系統中所有檔案的 metadata 都使用隨機密鑰進行加密，該密鑰是在首次安裝 iOS 或使用者清除裝置時製作而成。檔案系統密鑰則儲存在 Effaceable Storage 中。因為該密鑰儲存在裝置上，因此它不是用來維護資料的機密性，而是可以視需求快速清除(由使用者從「清除所有內容和設定」選項來清除，或者由使用者或管理者從行動裝置管理(MDM)伺服器、Exchange ActiveSync 或 iCloud 發出遠端清除指令來清除)。以此方式清除密鑰將會透過加密的方式讓裝置上的所有檔案無法取用。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.6.敏感性資料刪除

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 1.9

參考來源：

Android：How to Securely Delete All Data From Your Android Phone(查詢時間：2016/10/1)

iOS：iOS Security Guide(查詢時間：2016/10/1) P.10

CPI-04:受組織管理行動智慧裝置應充分運用行動作業系統提供資料刪除開關(Data Kill Switch)，刪除遺失或離職員工行動智慧裝置上的敏感資料。

說明：

為防止企業或個人行動智慧裝置遺失造成敏感性資料外洩，或回收離職員工個人行動智慧裝置中屬於的企業智慧財產權，Android 5.0 與 iOS 7.0 以後版本，均已具備資料刪除開關(Data Kill Switch)，當行動智慧裝置遺失或遭竊，可以由使用者自行登入 Android Device Manager(Android)或 iCloud 的 Find My iPhone(iOS)進行抹除，也只有使用者可以自行解鎖。企業受管理裝置的使用者可以透過行動裝置管理系統啟動資料刪除開關(參考 CPI-03)，一旦啟動清除，行動智慧裝置就無法復原資料及啟動手機。

需要注意的是 Android 需要與各品牌行動智慧裝置硬體進行整合，不像 iOS 為自有硬體及作業系統可自行整合外，各家推出具有此功能行動智慧裝置的機種與時程都不一，需要特別查閱細部規格。

安全開發生命週期：部署維運階段階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.6.敏感性資料刪除

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG1.13

參考來源：

Android：How to set up your smartphone killswitch (查詢時間：2016/10/1)

iOS：The kill switch is here: iOS 8 enables it by default (查詢時間：2016/10/1)

CPI-05:行動應用程式納入 CPI-04「資料刪除開關」時，應實作需經強認證要求始能啟動此功能，以避免遭受攻擊或濫用危及資料安全。

說明：

應用程式開發人員可能要納入應用程式特定的“資料刪除開關”到他們的行動應用App，為避免這樣的功能被濫用或遭惡意攻擊者利用，在需要時讓每應用刪除其應用程式的敏感資料執行強認證(參考 CPL-01，依美國 NIST 定義等級 3 以上的雙因素認證)要求，以保護資料安全。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.6.敏感性資料刪除

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 1.14

參考來源：N/A

CPI-06:當行動應用 App 被反安裝時，應依 CPI-02 設計一併移除敏感性資料，並通知雲端服務商或企業資料中心端刪除使用者資料或同意保留資料。

說明：

如依照 CPF-03 實作敏感性資料應該都會只儲存 Android 及 iOS 本地端的沙箱中，在行動應用 App 反安裝同時應一併被刪除，但如 CPF-04 設計與實作，敏感性資料將會以儲存位於雲端服務商或企業資料中心的資料庫或儲存服務(後端)為主，當使用者啟動反安裝程序前，應提示使用者同意刪除或是保留，並將訊息傳送至後端處理。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.2.6.敏感性資料刪除

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：CSA MAST 3.6

參考來源：N/A

3.1.10. 付費資源使用(J)

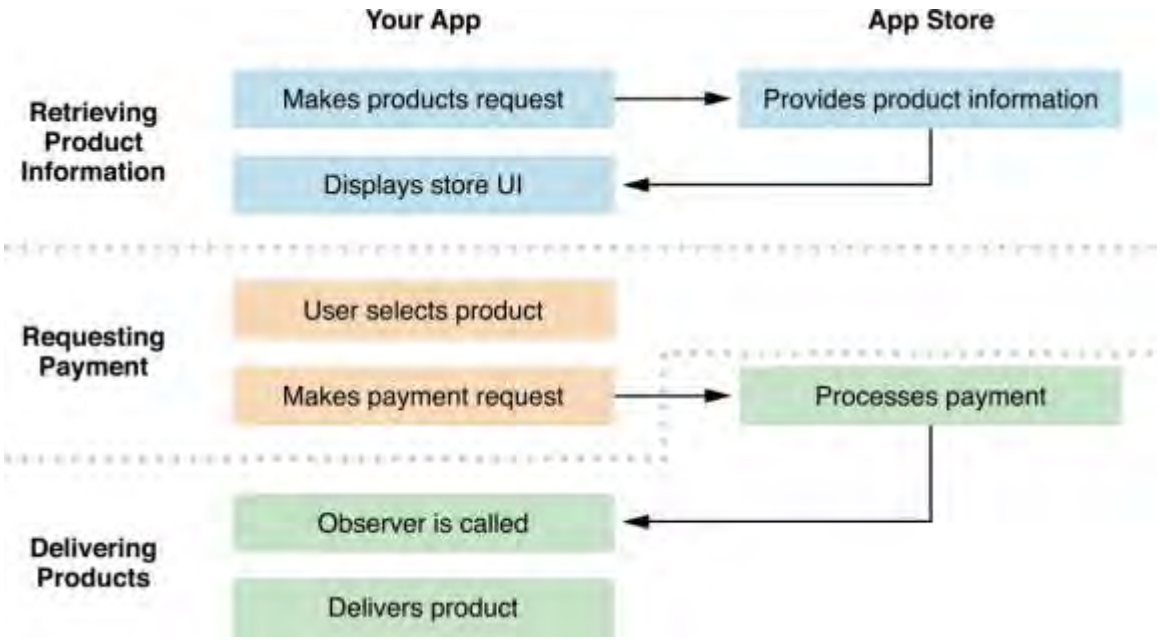
CPJ-01:於行動應用 App 內執行付費指示前，是否主動通知使用者，其資訊至少包含付費資源名稱、數量、金額及付費方式，並實作提示使用者同意及拒絕選項。

說明：

Android 開發人員可以在行動應用 App 內呼叫 In-app Billing API，向 Google Play 發送一個商品資訊，並且快速回覆儲存在商店付費資源名稱、數量、金額，商品資訊可在 Google Play Developer Console 上設定，可定義產品資訊如下：

- 唯一產品識別碼(Unique product ID，also called its SKU)。
- 產品分類(Product type)。
- 價錢(Pricing)。
- 產品描述(Description)。
- Google Play 產品處理及追蹤(handling and tracking of purchases for that product.)

iOS 則是可以運用 App Store 提供的 Store Kit framework 嵌入程式中，並至 itune connect 設定商品資訊，App 內購買流程如下圖：詳細參考 [In-App Purchase Programming Guide（查詢時間：2016/10/1)]



在正式執行付費指示前，需實作提示使用者同意及拒絕選項。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.3.1.付費資源使用

行動應用 App 基本資安檢測基準：

4.1.3.1.1.檢查行動應用程式內於付費前，是否主動通知使用者，且資訊至少包含付費資源名稱、數量、金額及付費方式。

4.1.3.1.2.(1)檢查行動應用程式內於付費時，是否提供使用者拒絕付費之選項。

4.1.3.1.2.(2)檢查在使用者拒絕付費的情況下，行動應用程式是否未進行付費。

各國開發要點參考：ENISA SSDG8.6

參考來源：

Android：Google Play Developer Console(查詢時間：2016/10/1)

iOS：In-App Purchase Programming Guide(查詢時間：2016/10/1)

CPJ-02:行動應用 App 不應有未向使用者明示且未經使用者同意，擅自呼叫行動智慧裝置通信功能，造成使用者費用損失的行為，包括在使用者無確認情況下撥打電話、發送簡訊、發送多媒體簡訊、開啟行動通信網路 連接並收發資料的行為。

說明：

行動智慧裝置會出現未向使用者明示且未經使用者同意，擅自呼叫行動智慧裝置通信功能，造成使用者費用損失的行為，較常出現在製造商、銷售商與電信服務商預置或上傳行動應用 App，此類軟體，可依國家通訊傳播委員會「105 年智慧型手機內建軟體資通安全檢測技術規範」(草案)要求進行檢測。

由企業或行動應用開發商開發的第三方行動應用 App 實作調用撥打電話、發送簡訊、發送多媒體簡訊、開啟行動通信網路 連接並收發資料的行為，需與 CPA-02 所宣告隱私權原則一致。不應有未向使用者明示且未經使用者同意，在背景有實作前述調用行動智慧裝置資源，造成使用者費用不預期流失。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.3.1.付費資源使用

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：YD/T 2407-2013 5.5.4.3

參考來源：N/A

CPJ-03:行動應用 App 開發人員應實作節約使用行網路流量最佳實務，如快速休眠、暫存檔等，以盡量減少對基地台的訊號負荷。

說明：

多種因素影響在設備上執行網路操作所需的能源量。 行動通訊網路活動需要比通過 Wi-Fi 活動顯著需要更多的能量。 在訊號差或不穩定訊號條件下，可能會導致更慢或有問題的交易，必須重試。低網路頻寬行動通訊網路不得不留在更長的時間來完成交易。甚至地理位置和行動通訊供應商的選擇會影響能量消耗，因為訊號條件和吞吐量是隨時變化。

行動應用 App 開發人員可以參考 3GPP Release 綠能技術文件（查詢時間：2016/10/1)實作節能最佳實務如快速休眠、暫存檔等，以盡量減少對基地台的訊號負荷。

另如偵測到行動應用 App 已移至背景運作，可以關閉該 App 對行動通訊基地台通訊。

在 iOS 實作可以參考 Energy Efficiency Guide for iOS Apps（查詢時間：2016/10/1)

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.3.1.付費資源使用

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 8.7

參考來源：

iOS：Energy and Networking(查詢時間：2016/10/1)

3GPP Knowledge Base

3GPP Green activities Energy Saving 20120924.zip(查詢時間：2016/10/1)

3.1.11. 付費資源控管(K)

CPK-01:有付費資源之行動應用 App 需要執行 CPJ-01 同意選項，需經身分認證後始能呼叫付費 API。

說明：

行動應用 App 如需使用讓使用者於 App 內購買或訂閱產品或服務需要綁定付費 API，在 Android 需要呼叫 Google Play Service 的 IInAppBillingService 的 API，呼叫 API 需要認證使用者端的憑證簽章及 API 的密鑰，實作請參考 Authenticating Your Client。(查詢時間：2016/10/1)

在 iOS 中 App 的付費將會以呼叫 Store Kit 的 API 提出付款請求，Store Kit 將會導至 App Store，必須經身分認證，才能完成交易。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.3.2.付費資源控管

行動應用 App 基本資安檢測基準：

4.1.3.2.1.檢查行動應用程式於付費時，是否提供身分認證機制。

各國開發要點參考：ENISA SSDG 8.4

參考來源：

Android：Implementing In-app Billing(查詢時間：2016/10/1)

iOS：In-App Purchase Programming Guide(查詢時間：2016/10/1)

CPK-02:行動應用 App 完成付費後，應提示該次付費資訊，至少包含付費資源名稱、付費時間及付費金額。

說明：

行動應用 App 在付費後會由應用程式商店，以電子郵件寄送消費明細給使用者，如 Android 平台 Google Play 發出的消費明細包含了「商家名稱」、「訂單編號及日期」、「付費資源名稱」及「付費金額」，樣本如下圖：

iOS 寄給使用者的電子郵件收據也包含了類似資訊。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.3.2.付費資源控管

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：N/A

CPK-03:行動應用 App 應提供安全查詢交易紀錄之管道，如在行動應用 App 選單內或於行動應用商店，且交易紀錄至少包含付費資源名稱、付費時間及付費金額之記錄。

說明：

行動應用商店均有保留使用者消費紀錄，使用者可以查詢到所有曾付費之付費資源名稱、付費時間及付費金額之記錄，這些紀錄包含取消及被刪除的 App。行動應用商店消費紀錄查詢網址如下：

Android:Google Play(https://play.google.com/store/account(查詢時間：2016/10/1))

iOS:App Store https://phobos.apple.com/WebObjects/MZFinance.woa/wa/purchaseHistory(查詢時間：2016/10/1))

上述網站均需經身分認證，以 TLS 加密將消費紀錄，回傳到使用者瀏覽網頁介面。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.3.2.付費資源控管

行動應用 App 基本資安檢測基準：

4.1.3.2.2.檢查行動應用程式於付費後，是否提供查詢交易紀錄之管道，且交易紀錄至少包含付費資源名稱、付費時間及付費金額之記錄。

各國開發要點參考：ENISA SSDG 8.1，8.5 ,CSA MAST

參考來源：N/A

CPK-04:行動應用 App 於執行 CPK-01 時如偵測到異常重複認證，如位置顯著發生變化時，使用者語言的變化等，應增加額外驗證措施。如以電子郵件或簡訊發送交易驗證碼至綁定行動智慧裝置。

說明：

為防止使用者付費資源被盜用，除了行動應用平台內建的非法活動偵測引擎外，來打擊詐欺行為，行動應用 App 的後端伺服器可以提供額外的資訊來提高偵測成功率，例如：一個遊戲的 App 提供會員購買金幣以加速遊戲進度，正常情況是會由不同會員在不同的地理位置及時間購買金幣，但從伺服器的交易紀錄卻發現同一會員使用者，在短時間內，在不同國家的行動應用商店均有下單紀錄，合理懷疑是會員帳戶被盜用。此類異常行為無法被行動應用商店偵測，只能由行動應用 App 開發人員，由會員基本資訊及交易資訊去關聯。

行動應用 App 可以創建與行動應用商店不同識別使用者名稱(Username)，並由此使用者名稱(或稱會員識別 ID)產生隨機但獨立的識別 ID，僅用於行動應用 App 及後端資料庫識別使用者，當後端資料庫交易紀錄關聯分析，如偵測到使用者習慣位置顯著發生變化時，使用者語言的變化或第一次偵測到使用新的行動應用裝置發出付費請求等，應實作增加額外驗證措施，如以電子郵件或簡訊發送交易驗證碼或使用兩階段驗證至綁定行動智慧裝置認證機制，由使用者確認後再呼叫付費 API。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

iOS：In-App Purchase Programming Guide:Detecting Irregular Activity (查詢時間：2016/10/1)

行動應用 App 基本資安規範：4.1.3.2.付費資源控管

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 8.2

參考來源：N/A

CPK-05:行動應用 App 移除時，應提示使用者未來仍持續線上支付訂閱資源，應預設取消或供使用者選擇取消或保留選項。

說明：

參考 CPI-06 當行動應用 App 由使用者執行反安裝時或刪除，會先發訊息至後端伺服器，伺服器會再向行動應用商店查詢是否仍有持續在未來支付訂閱產品。如有，應實作發訊息給行動應用 App 提示使用者取消線上支付或或供使用者選擇取消或保留選項。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.3.2.付費資源控管

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：CSA MAST 3.5

參考來源：N/A

CPK-06:行動應用 App 取用 Bar Code 或 QR Code 進行下一步交易動作需與用戶確認。

說明：

參考 CPK-01 實作當行動應用 App 取用 Bar Code 或 QR Code 進行下一步交易動作需與用戶確認提示彈跳視窗。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.3.2.付費資源控管

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：N/A

3.1.12. 使用者身分認證與授權(L)

CPL-01:行動應用 App 應設計並實作適當身分認證機制，並依使用者身分授權，以防止敏感資料被非授權人員存取。

說明：

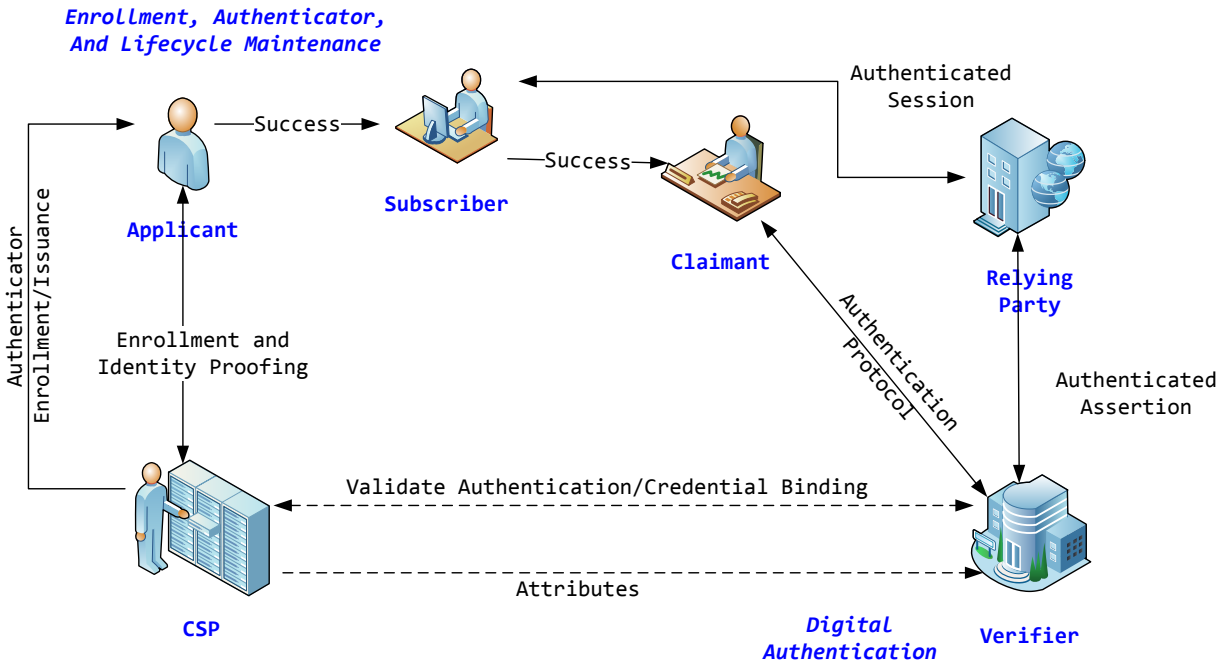
行動應用 App 蒐集、利用、儲存、分享、傳輸及刪除敏感性資料(CPD-01~CPI-02)，取用行動智慧裝置攝影鏡頭、麥克風(錄音)、感應器等本機資源，發起付費流程與 App 商店支付服務互動等對使用者隱私及財務都有直接影響，在實作 App 運行上述活動時，設計、開發實作人員需以「使用者意願」及「最小權限」為主要安全原則，在 App 運行先後順序上以「1.識別使用者身分」、「2.徵詢使用者意願(企業存取政策)」及「3.取得權限」，所以使用者身分認證是行動應用 App 取用敏感性資料權限的必要機制。

美國 NIST SP 800-63 將認證方法由較不安全到較安全分成四個等級：

- 等級 1：基本認證。但不可將密碼使用明碼的方式傳送到遠端。
- 等級 2：遠端網路認證。要有一定的方式確認對方的身分。要能避免竊聽、重送、線上猜密碼的攻擊。
- 等級 3：一定要用兩種以上的因子(Two-Factor)做認證 。

  三種單一因子：知道的資訊 (What you know?)、持有的 Token 裝置(What you have?)及生物特徵 (Who you are?)
- 等級 4：除了符合等級 3 一定要用符合 FIPS 140-2 硬體證明(Token)。

2016 年最新版 NIST SP 800-63-3 草案，由 NIST SP 800-63-2 大幅升級改組，由 NIST SP 800-63-3 DRAFT NIST Special Publication 800-63-3 Digital Authentication Guideline(頂層文件)、NIST SP 800-63-3A(Enrollment and Identity Proofing Requirements)、NIST SP 800-63-3B(Authentication and Lifecycle Management)及 NIST SP 800-63-3C(Federation and Assertions)等 4 份文件組成，共同構成數位認證模式(Digital Authentication Model)如下圖：



資料來源：[NIST SP 800-63-3 (查詢時間：2016/10/1)]

常見行動應用 App 身分認證方式：

- 行動智慧裝置解鎖 PIN 碼。
- 行動智慧裝置內建生物辨識(指紋、虹膜、面部、聲紋)。
- 雲端服務認證：Apple ID、Google ID、Microsoft Live ID。
- 開放式認證：OAuth 2.0。
- App 自建私有身分認證。
- 企業整合認證：Microsoft AD、Novell LDAP。

常見實作的技術：

- 帳號/密碼。
- 觸控螢幕滑動手勢。
- PKI。
- 多因素認證。
- 生物辨識。

現行應用在一般使用者最常運用的認證機制為 OAuth 2.0，使用者可以使用 Google、Facebook 及 Windows Live。

以上機制行動應用 App 設計及開發人員應視法規及實務需求，設計且實作適當身分認證機制。

安全開發生命週期：設計階段、開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

Android：Java

```
• //定義使用 ACCOUNT_MANAGER 與 INTERNET 權限
  <manifest ... >
      <uses-permission android:name="android.permission.ACCOUNT_MANAGER" />
      <uses-permission android:name="android.permission.INTERNET" />
      ...
  </manifest>
  //實作 CallBack
  AccountManager am = AccountManager.get(this);
  Bundle options = new Bundle();
  am.getAuthToken(
      myAccount_,                     // Account retrieved using getAccountsByType()
      "Manage your tasks",            // Auth scope
      options,                        // Authenticator-specific options
      this,                           // Your activity
      new OnTokenAcquired(),          // Callback called when a token is successfully
  acquired
      new Handler(new OnError()));    // Callback called if an error occurs

• private class OnTokenAcquired implements AccountManagerCallback<Bundle> {
      @Override
      public void run(AccountManagerFuture<Bundle> result) {
          ...
          Intent launch = (Intent) result.getResult().get(AccountManager.KEY_INTENT);
          if (launch != null) {
              startActivityForResult(launch, 0);
              return;
          }
      }
  }
  //獲取 token
  private class OnTokenAcquired implements AccountManagerCallback<Bundle> {
      @Override
      public void run(AccountManagerFuture<Bundle> result) {
```

```
            // Get the result of the operation from the AccountManagerFuture.
            Bundle bundle = result.getResult();

            // The token is a named value in the bundle. The name of the value
            // is stored in the constant AccountManager.KEY_AUTHTOKEN.
            token = bundle.getString(AccountManager.KEY_AUTHTOKEN);
            ...
        }
    }
    //呼叫 OAuth2 服務
    URL url = new URL("https://www.googleapis.com/tasks/v1/users/@me/lists?key=" +
    your_api_key);
    URLConnection conn = (HttpURLConnection) url.openConnection();
    conn.addRequestProperty("client_id", your client id);
    conn.addRequestProperty("client_secret", your client secret);
    conn.setRequestProperty("Authorization", "OAuth " + token);
```

iOS：參考 CPF-13 安全範例：使用 OAuth2。

行動應用 App 基本資安規範：4.1.4.1.使用者身分認證與授權

行動應用 App 基本資安檢測基準：

4.1.4.1.1.如行動應用程式存取與個人資料相關之敏感性資料，檢查行動應用程式是否提供認證機制。

4.1.4.1.2. 如行動應用程式存取與個人資料相關之敏感性資料，檢查行動應用程式是否提供身分授權機制。

各國開發要點參考：JSSEC AASDSCG .5.1

參考來源：

Android：

Android Authentication(查詢時間：2016/10/1)
https://developer.android.com/training/id-auth/authenticate.html (查詢時間：2016/10/1)

iOS：

Designing Secure User Interfaces(查詢時間：2016/10/1)
NIST SP 800-63(查詢時間：2016/10/1)
https://github.com/nxtbgthng/OAuth2Client(查詢時間：2016/10/1)

CPL-02:如使用固定帳號密碼作為唯一身分認證因素，視資料敏感性設定密碼強度，密碼安全性設計選用原則：

(a)在人類可記憶情況，視處理資料敏感程度不應少於 8~12 位。若搭配交易驗證碼使用則不應少於 4 位。

(b)建議採英數字混合使用，且宜包含大小寫英文字母或符號。

(c)不應訂為相同的英數字、連續英文字或連號數字，預設密碼不在此限。

(d)密碼與使用者帳號不應相同。

(e)密碼連續錯誤達 5 次，應鎖定或間隔 15 分鐘以上時間解鎖。

(f)變更密碼不得與前一次相同。

(g)使用預設密碼首次登入時，應強制變更預設密碼。

(h)密碼最長有效期限。

說明：

參考 CPL-01 依 NIST SP 800-63 說明，單純使用帳號密碼為安全度最低的身分認證方式，因其技術實作簡易，每部行動智慧裝置都可以在觸控介面上輸入密碼與遠端的伺服器認證。使用者為了方便記憶在應用系統或 App 沒有密碼強度的安全政策限定下，經常會選擇像 1234 或 abcd 這類弱密碼，易遭密碼暴力破解攻擊。

設定合理的密碼最小長度、增加密碼複雜性、輸入錯誤鎖定，都是常防止此類攻擊常見方法。

並可考量產業安全標準要求，例如：「金融機構辦理電子銀行業務安全控管作業基準」密碼錯誤 5 次，則鎖住帳號，不得再繼續執行交易。

密碼強度政策，以 Google 為例：

Google 帳戶密碼的長度不得少於 8 個字元，當中可包含：

- 大小寫英文字母：密碼有大小寫之分，因此系統會將「G」和「g」視為不同的字元。
- 數字。
- 以下符號：! " # $ % & ‘ ( ) * + , - . / : ; < = > ? @ [\ ] ^ { | } ~。
- 空格：密碼的開頭或結尾不得為空格，但其他部分可以使用空格。

請注意，您無法重複使用前一年使用過的密碼。此外，系統也不接受容易破解的密碼，例如：「12345678」。

進一步了解如何設定安全強度高的密碼(查詢時間：2016/10/1)。

安全開發生命週期：設計階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.4.1.使用者身分認證與授權

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考： ENISA SSDG 2.6,2.8,4.1

參考來源：iOS：

Apple Secure Coding Guide: Use reasonable pasword policies.(查詢時間：2016/10/1)

OWASP Authentication General Guidelines(查詢時間：2016/10/1)

CPL-03:行動應用 App 應提供使用者安全變更密碼功能

說明：

參考 CPL-02 行動應用平台或 App 為加強使用者密碼安全，落實執行密碼政策設定密碼最長有效期限，屆期使用者需更改密碼，另外常的情況是使用者的密碼可能是忘記或是已遭破解，需要立即更改密碼加強安全性。以單純使用帳號/密碼為唯一身分認證機制，需要謹慎設計、開發實作安全的密碼變更流程。

以變更 Google 密碼(查詢時間：2016/10/1)為例，如僅為變更密碼，使用者需登入 Google account 網頁，輸入原有密碼，才能重設密碼，如為忘記密碼時，會寄送到 Gmail 信箱或其他備援電子郵件到使用者電子郵件信箱或行動電話一次性暫時 url 連結，依程序進行重設，全程都在 TLS 加密連線狀態下執行，以保護帳號密碼被非授權進行身分授權劫奪。

安全開發生命週期：設計、開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.4.1.使用者身分認證與授權

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 2.4

參考來源：N/A

CPL-04:避免在行動應用 App 中直接使用設備 ID 作為使用者身分追蹤識別作為唯一因素，建議以帳號及密碼作為主要因素，設備 ID 可作為多因素認證的其他因素。

說明：

雖然作業系統有提供唯一裝置識別碼，但依據保護個人資料與資料使用的規則，不建議繼續使用此資料當作 APP 的主要識別碼。

在 iOS 部分：目前此裝置識別碼已被保護，程式碼已經抓不到正確的裝置碼，且每次安裝時，取得的資料，就會改變一次。雖然仍可被部分使用，但實際上沒有唯一性，只可被當作參考用途。

Android 部分：雖然可以取得 SIM card 的唯一碼，但仍不建議單獨使用它為認證資料。因為如果設備可能沒有 SIM card 只有 WiFi，那就無法取得唯一碼。另外 SIM card 的唯一碼是 IMSI,也可能會被偽裝，無法完全信任。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

Android：

請注意，Android 環境中目前無任何設備提供的 ID 是無法被偽裝的，所以不可以使用任何 MAC Address, IMSI, IMEI, UUID, android id, device id 這些有設備詢問而來的 ID 做為授權的根據

iOS：Objective-C

Device.UUID 範例碼

```
[NSString stringWithFormat:@"UUID=%@", [[[UIDevice currentDevice]
identifierForVendor] UUIDString]];
```

行動應用 App 基本資安規範：4.1.4.1.使用者身分認證與授權

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 4.6

參考來源：CWE/OWASP：

M5 - Poor Authorization and Authentication(查詢時間：2016/10/1)

CWE 200(查詢時間：2016/10/1)

CPL-05:行動應用程式在處理敏感性資料時可實作多因素認證，加強身分認安全性，除帳號密碼外作為主要因素外，可考慮使用一次性密碼 token(OTP Token)、安全元件(Secure Element)、雙因素驗證、生物特徵(聲紋、指紋、臉部識別)、知識詢問及約定資訊等。

說明：

在某些使用情境下帳號(ID)和密碼無法提供用於行動應用 App 足夠的安全性。當存取敏感性資料或實施付費交易，可以實施雙因素身分驗證。雖然每次登入實施雙因素身分認證過於繁瑣實務上可能不是可行的，但是可以在每隔一段存取所選擇的功能時，或可以使用在需要存取更敏感資訊時，並可以作為驗證的第二層。

一些設備和附加允許開發人員使用例如：一個安全元素有時通過 SD 卡模塊-提供此功能的設備的數量可能增加。開發人員應該利用這種能力來儲存密鑰、證書和其他敏感資料。使用通過 FIPS 140-2 3 認證的 SD 卡作為安全元件來提供安全認證，雖然可能的第二因子的標準加密的 SD 卡一個更高的安全水準，但是不推薦，因為當一旦插入並固定後，它變成該裝置的一個為不可分割的一部分。

如有可能，可以考慮使用其他身分驗證因子應用提供存取敏感資料或介面在可能的情況，例如：聲紋、指紋、臉部識別。

安全開發生命週期：設計、開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.4.1.使用者身分認證與授權

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 4.5, ENISA SSDG 2.3

參考來源：CWE/OWASP：

M5 - Poor Authorization and Authentication (查詢時間：2016/10/1)

CWE 308 (查詢時間：2016/10/1)

CPL-06:行動應用 App 存取敏感性資料或是呼叫行動智慧裝置資源，如攝影鏡頭、麥克風等有地域或網段的限制，可增加 GPS 座標及網段範圍符合作為認證因子之一。

說明：

在企業或是政府組織的應用，有依場所地理位置限制敏感性資料存取需求，可在實作敏感性資料存取時，將行動智慧裝置地理位置資訊(GPS 座標範圍)或取得內部網段 IP 作為認證因子之一，如某些敏感性資料存取須在內部網路某網段才能存取，如果不符合地理位置或 IP 範圍，伺服器可以中斷行動使用者終端連線。

Android 取用地理位置資訊實作可以參考:Location Strategies(查詢時間：2016/10/1)

iOS 取用地理位置資訊實作可以參考 Location and Maps Programming Guide(查詢時間：2016/10/1)

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

Android：Location Strategies(查詢時間：2016/10/1)

iOS：Location and Maps Programming Guide(查詢時間：2016/10/1)

行動應用 App 基本資安規範：4.1.4.1.使用者身分認證與授權

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 4.4

參考來源：N/A

CPL-07:行動應用 App 如使用滑動可視密碼(Swipe-based)作為密碼，易遭塗抹的攻擊(smudge-attacks)，在設計上應導入允許重複圖案措施因應。

說明：

使用者可能選擇「滑動(Swipe-based)可視密碼」作為行動智慧裝置密碼解鎖密碼，因為使用者在多次使用後觸控螢幕後會留下手指的油漬，如果使用者沒有經常去擦拭清潔，當行動智慧裝置遭竊或遺失，如果「滑動可視密碼」沒有允許重複圖案輸入，在惡意人士可以施行「塗抹的攻擊」，由殘留油漬推算出使用者設定密碼軌跡，進而解鎖行動智慧裝置，造成敏感性資料外洩。

開發實作人員在實作「滑動可視密碼」，應導入允許重複圖案(以增加安全度、避免被此種攻擊)，使惡意人士無法使用「塗抹的攻擊」解鎖行動智慧裝置。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.4.1.使用者身分認證與授權

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 2.7

參考來源：

Smudge Attacks on Smartphone Touch Screens, Adam J. Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M. Smith (查詢時間：2016/10/1)

CPL-08:行動應用 App 應運用作業系統平台提供內建安全機制，如沙箱應用程式及通過身分認證後，始能允許存取敏感性資料及行動裝置資源，如電話、簡訊、攝影鏡頭、GPS 及麥克風等。

說明：

參考 CPD-05 及 CPD-06，存取敏感性資源，應須經使用者身分認證及授權，並優先呼行動應用平台提供 API 來調用行動裝置資源，如電話、簡訊、攝影機、GPS、麥克風等。

Android 實作請參考：Android Interfaces and Architecture(查詢時間：2016/10/1)

iOS 實作參考：存取媒體層函式庫

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.4.1.使用者身分認證與授權

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考

YD/T 2407-2013 5.6.2,NIST 800 SP-163 3.1.2, JSSEC AASDSCG .5.2

參考來源：N/A

3.1.13. 連線管理機制(M)

CPM-01:行動應用 App 實作 CPG-02 TLS 連線，需使用編碼長度為 128 位元(含)以上之交談識別碼(Session ID)。

說明：

攻擊者可以在經過攫取分析一系列的 ID 值，破解交談識別碼。交談識別碼必須足夠長，以防止暴力攻擊，並驗證有效交談的存在。交談識別碼長度必須至少為 128 位(16 字節)。128 位元的交談識別碼長度被設置為基於在“交談識別碼熵(Entropy)64 位元，為交談識別碼的一半(CPM-02)”所作的假設的參考。然而，該數目不應該被認為是絕對的最小值，作為其他實作因素可能影響其強度。例如：有公認良好的的實作方式，如 Microsoft ASP.NET 中，利用 120 位元的隨機數為它的交談識別碼，可以提供很好的有效熵，和(20 字符的字符串表示)其結果，可以認為足夠長的時間，以避免猜測或暴力攻擊。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.4.2.連線管理機制

行動應用 App 基本資安檢測基準：

4.1.4.2.1.(1)檢查行動應用程式是否採用有效長度為 128 位元(含)以上之交談識別碼。

各國開發要點參考：N/A

參考來源：OWASP Session Management Cheat Sheet:Session ID Length(查詢時間：2016/10/1)

CPM-02:行動應用 App 連線使用交談識別碼，應不運用可預測規則種子產生亂數。

說明：

在攻擊者能夠猜測或通過統計分析技術預測有效的 Session ID。為防止此類猜測攻擊。交談識別碼必須是不可預知的(足夠隨機)。

為了達成這個目的，必須使用一個良好的 PRNG(虛擬隨機數產出器)。 交談識別碼值必須提供熵的至少 64 位元(如果使用了良好的 PRNG，這個值被估計為所述交談識別碼的長度的一半)。

交談識別碼熵確實受其他外部和難以測量的因素，如併發活動交談的數目的網路應用程式通常具有的絕對交談有效期逾時，每秒攻擊交談識別碼猜測的量可以使和目標 web 應用程式可以支援等。如果使用具有 64 位元的熵 session ID，它會採取一個攻擊者至少 292 年成功猜出有效的交談識別碼，假設攻擊者可以嘗試每 10,000 個猜測擁有 10 萬有效同時提供 Web 應用程式 Sessions。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.4.2.連線管理機制

行動應用 App 基本資安檢測基準：

4.1.4.2.1.(2) 檢查行動應用程式使用之交談識別碼是否未與時間、使用者上傳資料、具規則性之數字或字串有直接關聯或難以偽造。

各國開發要點參考：ENISA SSDG 4.3

參考來源：OWSP Session Management Cheat Sheet：Session ID Entropy（查詢時間：2016/10/1)

CPM-03:行動應用 App 連線使用交談識別碼，應實作具備逾時失效(Session time-out)機制。

說明：

由於行動智慧裝置經常丟失或被盜，並且攻擊者可能利用一個應用程式來存取敏感資料，執行交易或研究設備所有者的帳戶。尤其是銀行或交易類的應用程式。建議行動應用 App，在登入後也進行時間控管，以加強安全性。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

Android：Java

可以在 Activity 內加上下面二個方法來達成 local-session-timeout

```
@Override public boolean dispatchTouchEvent(MotionEvent ev) {
        lastActivity = new Date().getTime();  //取得最後動作的時間
        return super.dispatchTouchEvent(ev);
}
@Override public void onResume() {
        long now = new Date().getTime();
        if (now - lastActivity > 300000) {     //超過五分鐘則自動登出
        // startActivity and force logon     } }
```

iOS：Objective-C

- *iOS: local-session-timeout code 實作*
- *1.啟動時間計算*

```
-    [myApp setLoginDate:[NSDate date]];
- - (void) setLoginDate: (NSDate *)indate
- {
-    loginDate = indate;
- }
```

- *2.檢查是否在安全時間內*

```
- [NSString stringWithFormat:@"isValidDate=%d",[myApp isLoginDateValid],
- - (BOOL) isLoginDateValid
- {
-    if (!loginDate)
-        return FALSE;
-    //valid for 60 seconds
-    NSDate *max_date = [loginDate dateByAddingTimeInterval:60];
-    return
```

```
        (([loginDate compare:[NSDate date]] == NSOrderedAscending) &&
        ([max_date compare:[NSDate date]] == NSOrderedDescending));
}
```

行動應用 App 基本資安規範：4.1.4.2.連線管理機制

行動應用 App 基本資安檢測基準：

4.1.4.2.1.(3)檢查行動應用程式使用之交談識別碼是否具備逾時失效機制。

各國開發要點參考：N/A

參考來源：CWE/OWASP：

M9 - Improper Session Handling(查詢時間：2016/10/1)

CWE 613(查詢時間：2016/10/1)

CPM-04:行動應用 App 使用伺服器憑證需仍於有效期間內、未被註銷(Revoke)，且憑證之主體名稱與主體別名包含連線之伺服器網域名稱亦需有效。

說明：

在使用憑證的前提之下，即 HTTPS 的通訊架構下，系統元件有提供檢查憑證的有效期間、是否註銷、根憑證未信任等機制。

強制接受錯誤的憑證是不安全的程式開發實務，必須有正確的憑證，並且可只允許某個網域名稱，才是安全的實務。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.4.2.連線管理機制

行動應用 App 基本資安檢測基準：

4.1.4.2.2.(1)檢查行動應用程式是否使用伺服器憑證仍於有效期間內、未被註銷(Revoke)，且憑證之主體名稱與主體別名包含連線之伺服器網域名稱。

各國開發要點參考：N/A

參考來源：

Android：Security with HTTPS and SSL（查詢時間：2016/10/1)

iOS：

NSURLSession server trust evaluation - apple develper（查詢時間：2016/10/1)

Appendix A: Common Server Trust Evaluation Errors - apple developer（查詢時間：2016/10/1)

CPM-05:行動應用程式應使用憑證綁定(Certificate Pinning)方式驗證並確保連線之伺服器為行動應用程式開發人員所指定。

說明：

在全球漏洞資料庫網站揭露一個 CVE(Common Vulnerabilities and Exposures)漏洞(編號：CVE-2014-6693)。CVE 漏洞報告指出，因為行動應用 App 沒有驗證 x.509 的 CA(Certificate Authority)憑證，駭客可以藉此發動中間人攻擊，以竊取使用者敏感性資料。

開發人員應使用憑證綁定(Certificate Pinning)的方式，把需要比對的信任發行者發行憑證預先存放在 App 裡，指定特定 Domain 就只能使用特定憑證，等到要進行 SSL Handshake 的時候，再與伺服器的憑證進行比對。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

Android：Java

- 假設你有由知名 CA 頒發的證書的 Web 伺服器，你可以用簡易程式碼安全要求：

```
• URL url = new URL ( "https://wikipedia.org" );
• URLConnection urlConnection = url . openConnection ();
• InputStream in = urlConnection . getInputStream ();
• copyInputStreamToOutputStream ( in , System . out );
```

- 如果你想客製設定 HTTP 請求，可以改使用 HttpsURLConnection 或是強制轉型為 HttpURLConnection

```
public class KeyPinStore {

    private static KeyPinStore instance = null;
    private SSLContext sslContext = SSLContext.getInstance("TLS");

    public static synchronized KeyPinStore getInstance() throws CertificateException,
IOException, KeyStoreException, NoSuchAlgorithmException, KeyManagementException{
        if (instance == null){
            instance = new KeyPinStore();
        }
        return instance;
    }
```

```
    private KeyPinStore() throws CertificateException, IOException,
KeyStoreException, NoSuchAlgorithmException, KeyManagementException{
        // Load CAs from an InputStream
        // (could be from a resource or ByteArrayInputStream or ...)
        CertificateFactory cf = CertificateFactory.getInstance("X.509");
        // randomCA.crt should be in the Assets directory
        InputStream caInput = new
BufferedInputStream(MainActivity.context.getAssets().open("randomCA.crt"));
        Certificate ca;
        try {
            ca = cf.generateCertificate(caInput);
            System.out.println("ca=" + ((X509Certificate) ca).getSubjectDN());
        } finally {
            caInput.close();
        }

        // Create a KeyStore containing our trusted CAs
        String keyStoreType = KeyStore.getDefaultType();
        KeyStore keyStore = KeyStore.getInstance(keyStoreType);
        keyStore.load(null, null);
        keyStore.setCertificateEntry("ca", ca);

        // Create a TrustManager that trusts the CAs in our KeyStore
        String tmfAlgorithm = TrustManagerFactory.getDefaultAlgorithm();
        TrustManagerFactory tmf = TrustManagerFactory.getInstance(tmfAlgorithm);
        tmf.init(keyStore);

        // Create an SSLContext that uses our TrustManager
        // SSLContext context = SSLContext.getInstance("TLS");
        sslContext.init(null, tmf.getTrustManagers(), null);
    }

    public SSLContext getContext(){
        return sslContext;
    }
}
```

iOS：Objective-C

完整的 SSL Pinning 範例(以 https://www.owasp.org 與 https://gca.nat.gov.tw)

準備：取出 SSL 網站的公鑰

1.由網站獲取公鑰，以利 https ssl ceriticate 認證時比對

```
  ex +'/BEGIN CERTIFICATE/,/END CERTIFICATE/p' <(echo | openssl s_client -showcerts -connect www.owasp.org:443) -scq > file.crt
  ex +'/BEGIN CERTIFICATE/,/END CERTIFICATE/p' <(echo | openssl s_client -showcerts -connect gca.nat.gov.tw:443) -scq > gca.crt
```

2.然後轉成 der

```
  openssl x509 -outform der -in owasp.crt -out owasp2.der
  openssl x509 -outform der -in gca.crt -out gca2.der
```

3.附加為專案內容，以用於比對

範例一：元件 NSURLConnectionDelegate

1.實作 NSURLConnectionDelegate

2.建立 NSURLConnection

```
   urlconnection = [[NSURLConnection alloc] initWithRequest:theRequest delegate:self];
```

3.監聽、驗證 SSL Pinning

```
- (BOOL)connection:(NSURLConnection *)connection canAuthenticateAgainstProtectionSpace:(NSURLProtectionSpace *)protectionSpace {
return [protectionSpace.authenticationMethod isEqualToString:NSURLAuthenticationMethodServerTrust];
}
//可在 didReceiveAuthenticationChallenge 進行憑證驗證
//或者實作 willSendRequestForAuthenticationChallenge 自行更早決定成功/失敗/驗證等
//於 willSendRequestForAuthenticationChallenge 可回傳以下幾種方式
// 1.useCredential:forAuthenticationChallenge:
// 2.continueWithoutCredentialForAuthenticationChallenge:
// 3.cancelAuthenticationChallenge:
// 4.performDefaultHandlingForAuthenticationChallenge:
// 5.rejectProtectionSpaceAndContinueWithChallenge:
- (void)connection:(NSURLConnection *)connection didReceiveAuthenticationChallenge:(NSURLAuthenticationChallenge *)challenge {
   //Enable Pinning
   if ([[[challenge protectionSpace] authenticationMethod] isEqualToString:NSURLAuthenticationMethodServerTrust]){
       do
```

```
        {
            SecTrustRef serverTrust = [[challenge protectionSpace] serverTrust];
            if(nil == serverTrust)
            {
                [self AppendLog:@"serverTrust is nil"];
                break; /* failed */
            }
            OSStatus status = SecTrustEvaluate(serverTrust, NULL);
            if(!(errSecSuccess == status))
            {
                [self AppendLog:@"SecTrustEvaluate status is errSecSuccess"];
                break; /* failed */
            }
            SecCertificateRef serverCertificate =
SecTrustGetCertificateAtIndex(serverTrust, 0);
           if(nil == serverCertificate)
            {
                [self AppendLog:@"serverCertificate is nil"];
                break; /* failed */
            }
            CFDataRef serverCertificateData =
SecCertificateCopyData(serverCertificate);
            if(nil == serverCertificateData)
            {
                [self AppendLog:@"serverCertificateData is nil"];
                break; /* failed */
            }
            const UInt8* const data = CFDataGetBytePtr(serverCertificateData);
            const CFIndex size = CFDataGetLength(serverCertificateData);
            NSData* cert1 = [NSData dataWithBytes:data length:(NSUInteger)size];
            if(nil == cert1)
            {
                [self AppendLog:@"server cert is nil"];
                break; /* failed */
            }
            NSString *file = [[NSBundle mainBundle] pathForResource:@"domain"
ofType:@"der"];
            NSData* cert2 = [NSData dataWithContentsOfFile:file];
```

```
            if(nil == cert1 || nil == cert2)
            {
                [self AppendLog:@"local pinning cert is nil"];
                break; /* failed */
            }
            const BOOL equal = [cert1 isEqualToData:cert2];
            if(!equal){
                [self AppendLog:@"certs not equal"];
                break; /* failed */
            }
            [self AppendLog:@"certs are GOOD"];
            // The only good exit point
            return [[challenge sender] useCredential: [NSURLCredential
credentialForTrust: serverTrust]
                                    forAuthenticationChallenge: challenge];

        } while(0);
    }
    // Bad dog
    return [[challenge sender] cancelAuthenticationChallenge: challenge];
}
```

- *範例二：元件 NSURLSession*
- *1.建立 NSURLSession，並啟動之 by [NSURLSessionDataTask]*

```
    NSURLSessionConfiguration *sessionConfig = [NSURLSessionConfiguration
defaultSessionConfiguration];
    urlsession = [NSURLSession sessionWithConfiguration:sessionConfig
delegate:self delegateQueue:nil];
    [[urlsession dataTaskWithURL:url completionHandler:^(NSData * _Nullable
data, NSURLResponse * _Nullable response, NSError * _Nullable error) {
        // response management code
        if (error)
        {
            [self AppendLog:[NSString stringWithFormat:@">> dataTask Error:%@",
error.description]];
        }
        else
        {
```

```
            [self AppendLog:[NSString stringWithFormat:@">> dataTask got data %lu", data.length]];
        }
    }] resume];
```

2.驗證SSL Pinning

```
-(void)URLSession:(NSURLSession *)session didReceiveChallenge:(NSURLAuthenticationChallenge *)challenge completionHandler:(void (^)(NSURLSessionAuthChallengeDisposition, NSURLCredential * _Nullable))completionHandler {
    // Get remote certificate
    SecTrustRef serverTrust = challenge.protectionSpace.serverTrust;
    SecCertificateRef certificate = SecTrustGetCertificateAtIndex(serverTrust, 0);
    // Set SSL policies for domain name check
    NSMutableArray *policies = [NSMutableArray array];
    [policies addObject:(__bridge_transfer id)SecPolicyCreateSSL(true, (__bridge CFStringRef)challenge.protectionSpace.host)];
    SecTrustSetPolicies(serverTrust, (__bridge CFArrayRef)policies);
    // Evaluate server certificate
    SecTrustResultType result;
    SecTrustEvaluate(serverTrust, &result);
    BOOL certificateIsValid = (result == kSecTrustResultUnspecified || result == kSecTrustResultProceed);
    // Get local and remote cert data
    NSData *remoteCertificateData = CFBridgingRelease(SecCertificateCopyData(certificate));
    NSString *pathToCert = [[NSBundle mainBundle]pathForResource:@"domain" ofType:@"der"];
    NSData *localCertificate = [NSData dataWithContentsOfFile:pathToCert];
    // The pinnning check
    if ([remoteCertificateData isEqualToData:localCertificate] && certificateIsValid) {
        NSURLCredential *credential = [NSURLCredential credentialForTrust:serverTrust];
        [self AppendLog:@"URLSession pinning completionHandler credential(GOOD)"];
        completionHandler(NSURLSessionAuthChallengeUseCredential, credential);
    } else {
```

```
        [self AppendLog:@"URLSession pinning completionHandler BAD(NULL)"];
        completionHandler(NSURLSessionAuthChallengeCancelAuthenticationChallenge
  , NULL);
    }
}
```

行動應用 App 基本資安規範：4.1.4.2.連線管理機制

行動應用 App 基本資安檢測基準：

4.1.4.2.2.(2)檢查行動應用程式是否使用憑證綁定(Certificate Pinning)方式驗證並確保連線之伺服器為行動應用程式開發人員所指定。

各國開發要點參考：N/A

參考來源：

Android：

https://www.owasp.org/index.php/Certificate and Public Key Pinning (查詢時間：2016/10/1)

iOS：

NSURLSession server trust evaluation - apple develper(查詢時間：2016/10/1)

Enforcing Stricter Server Trust Evaluation / certificate authority pinning - apple developer(查詢時間：2016/10/1)

Certificate and Public Key Pinning - owasp(查詢時間：2016/10/1)

CPM-06:行動應用 App 應使用憑證驗證鏈驗證各段連線跳躍(Hop)TLS 憑證為行動作業系統內建可信任之憑證機構、政府機關、企業所簽發公開憑證(Public CA)。

說明：

實作 TLS 連線，常見問題如下：

1.伺服器使用未由信任公開憑證授權中心(Certificate Authority)製發憑證，無法被驗證為合法憑證。

2.使用自簽憑證，無法被驗證為合法憑證。

3.伺服器憑證缺乏中間 CA 驗證，無法用憑證驗證鏈驗證為合法憑證。

在 App 端通常只會信任由可信任之憑證機構、政府機關、企業簽發公開憑證(Public CA)，因根 CA 並不直接簽發伺服器憑證，而是一層一層簽發下來，所以必須透過伺服器合法性需由憑證驗證鏈機制一路驗證至根 CA。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.4.2.連線管理機制

行動應用 App 基本資安檢測基準：

4.1.4.2.3.如行動應用程式使用安全加密傳輸協定，檢查行動應用程式是否驗證並確保伺服器憑證為行動作業系統內建可信任之憑證機構、政府機關、企業簽發。

各國開發要點參考：

ENISA SSDG 3.4,NIST 800 SP-163 3.1.4,NIST SP 800-57 Part1,Rev4

參考來源：N/A

CPM-07:行動應用 App 應於完成 CPM-06 與伺服器端憑證鏈驗證後，始能傳輸敏感性資料。

說明：

參考 CPM-06 憑證常見問題，如無法驗證憑證合法性，行動應用 App 就無法確認即將連線的伺服器端是否為可信任，有可能是已經遭受中間人攻擊，或遭連線劫奪至不明目的地的伺服器，造成敏感性資料外洩。

行動應用 App 在連線至伺服器端連線至無法驗證伺服器，應停止網路交握，並將不安全連線訊息警告使用者注意。直到驗證憑證合法之前，不能進行敏感性資料傳輸。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.4.2.連線管理機制

行動應用 App 基本資安檢測基準：

4.1.4.2.4.(1)如不符合檢測編號 4.1.4.2.2 或 4.1.4.2.3 之技術要求，檢查行動應用程式是否未與伺服器進行連線與傳輸敏感性資料。

4.1.4.2.4.(2)檢查行動應用程式是否使用符合檢測編號 4.1.4.2.2 與 4.1.4.2.3 之憑證與伺服器進行連線與傳輸敏感性資料。

各國開發要點參考：ENISA SSDG3.5,ENISA SSDG 4.2

參考來源：N/A

CPM-08:行動應用 App 的使用者介面應儘可能讓使用者容易查詢到憑證是否有效。

說明：

現今多數原生行動應用 App 未如網頁瀏覽器(https://)提供使用者於操作介面上檢視 TLS 連線憑證是否有效設計，使用者無從判斷是否為高風險連線行為。

開發人員應實作可顯示驗證憑證有效的資訊頁面。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.4.2.連線管理機制

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 3.6

參考來源：N/A

3.1.14. 防範惡意程式碼與避免資訊安全漏洞(N)

CPN-01:如有程式碼，使用可檢測行動應用 App 原始碼安全檢測工具或人工進行靜態分析，檢視權限並比對是否與 CPD-03 安全設計及使用者設定權限相符(permission mapping analysis)。

說明：

行動應用程式的安全性審查可透過程式碼檢測工具或人工程式碼檢視進行審查，此過程需按照 ISO/IEC 17025 標準進行，且需審查之項目應考量相關對應之標準或規範施行檢測。本指引建議提供安全需求項目與理想檢核結果並加以說明，在檢核過程中不強制規範使用靜態或動態方法或兩者並用，作為履行審核項目要求的方法組合。

- 一致性：行動應用程式審核結果應與本文中提到的要求結果一致。要求可能映射到兩個或更多的安全項目，當檢核的結果不明確時應將結果視為無效。
- 可比性：行動應用程式審核結果應該沒有差異，並與其他現有已知的審核計畫結果達成一致。行動應用程式審查結果應與在本指引中提到的結果符合。
- 重複性：當一個行動應用程式重複審核時，人工與工具之審核結果應該保持不變，而不應有差異，此過程應重複三次或更多次以確認其重複性結果。此外，同樣的檢測問題，應在兩個不同的行動應用程式檢測環境無結果差異。
- 測試人員執行行動應用 App 原始碼安全檢測，並以工具或人工進行靜態分析時，應檢視權限並比對是否與 CPD-03 安全設計及使用者設定權限相符。

安全開發生命週期：測試階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.5.1.防範惡意程式碼與避免資訊安全漏洞

行動應用 App 基本資安檢測基準：

4.1.5.1.1.(5)檢查行動應用程式是否未針對其他行動應用程式或行動作業系統之檔案，在未授權情況下，嘗試進行查詢、新增、修改、刪除、存取遠端服務、提權等行為。

各國開發要點參考：NIST SP 800-1163 3.1.2 ,CSA MAST 4.1.1

參考來源：N/A

CPN-02:如無程式碼使用「逆向工程法」輔以「自動化與人工原始碼分析」檢視權限並比對是否符合 CPD-03 安全設計及使用者設定權限相符。

說明：

測試人員使用「逆向工程法」輔以「自動化與人工原始碼分析」檢視權限並比對是否符合 CPD-03 安全設計及使用者設定權限相符。

當無法提供審核的原始程式碼時，仍應通過工具來進行靜態和動態審核測試，例如：iOS 應用程式檔案(IPAs)或 Android 應用程式包(APKs)。此方法應符合上述一致性，可比性和可重複性的要求。這個過程也需要符合 ISO/IEC 17025 的一致性規範。行動審核項目應根據指引所定義的安全項目進行靜態與動態檢核。

安全開發生命週期：測試階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.5.1.防範惡意程式碼與避免資訊安全漏洞

行動應用 App 基本資安檢測基準：

4.1.5.1.1.(5)檢查行動應用程式是否未針對其他行動應用程式或行動作業系統之檔案，在未授權情況下，嘗試進行查詢、新增、修改、刪除、存取遠端服務、提權等行為。

各國開發要點參考：NIST 800 SP-163 3.1.2, CSA MAST 4.1.2

參考來源：N/A

CPN-03:行動應用 App 應考量中央處理單元(CPU)、記憶體(Memory)、輸出入(Input/Output)單元能耗問題，找出個別程序、API 或行動智慧裝置服務呼叫過於損耗功率。

說明：

裝置功耗是開發與使用行動應用程式時常被忽視的問題之一，過大的功率消耗可能會降低行動裝置的可持續性。因此，開發人員應將功耗需求列為行動應用程式安全測試的要求之一。以下是安全要求與行動功耗與相關的審核準則：

CPU/Memory 使用率

•評估應用程式在待機模式或操作期間，其 CPU/Memory 的使用率是否超過平均值。

Input/Output 問題

•評估應用程式在待機模式或操作使用，其 I/O 使用率是否超過平均值。

測試人員如發現有能耗過度問題，應找出個別程序、API 或行動智慧裝置服務呼叫以修正滿足此一安全要求。

安全開發生命週期：測試階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.5.1.防範惡意程式碼與避免資訊安全漏洞

行動應用 App 基本資安檢測基準：

4.1.5.1.1.(6)檢查行動應用程式是否未包括可導致其他行動應用程式或行動作業系統，發生未預期錯誤、資源明顯耗損、重新啟動或關閉等行為。

各國開發要點參考：CSA MAST 4.2.8

參考來源：N/A

CPN-04:依正常及錯誤使用案例測試計畫，以動態測試行動應用 App 是否會發生未預期錯誤、資源明顯耗損、重新啟動或關閉。

說明：

設計一個正常及錯誤使用的案例測試計畫，以系統化方式執行行動應用 App 動態測試，觀察是否會發生發生未預期錯誤、資源明顯耗損、重新啟動或關閉。

測試人員應紀錄測試狀況，將測試異常結果回饋給開發實作人員。

安全開發生命週期：測試階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.5.1.防範惡意程式碼與避免資訊安全漏洞

行動應用 App 基本資安檢測基準：

4.1.5.1.1.(6)檢查行動應用程式是否未包括可導致其他行動應用程式或行動作業系統，發生未預期錯誤、資源明顯耗損、重新啟動或關閉等行為。

各國開發要點參考：N/A

參考來源：N/A

CPN-05:應檢測行動應用內部是否有非由使用者發起非預期與其他應用程式或第三方服務通訊或不明目的地等行為。

說明：

在 App 開發實作階段如果缺乏嚴格開發流程控管或審核機制，有可能會造成潛在隱私或敏感資料外洩。開發人員可惡意或過失將地理位置或相關資訊的轉移到不當目的地或其他應用程式，因為開發測試人員可能沒有在 App 的需求、設計及開發階段中考慮這些資訊的安全性。這種行為是不恰當且過多的資訊揭露是不良結果的軟體開發實務。建議可以用來評估應用程式開發的安全性如下：

1.評估如果 Android 版本有 App 內部活動保護機制。這個 App 內部活動應該專注於 Android 元件，例如：Activity, Service, BroadcastReceiver 及 Content Provider，或 iOS 元件，如 Container View。這些機制將有效實施防止無意洩露隱私或敏感資訊。

2.評估如果應用程式對 Android 元件，例如：Activities, Service and Content Provider 或 iOS 元件如 Container View 產生歸因保護機制。這些應在 App 執行階段來建立。這種保護機制應防止洩漏或歸因傳輸至不明目的地。

3.評估如果應用程式以這樣程式設計的方式，這將可能被利用逆向工程注入惡意程式碼，蒐集隱私或敏感資訊。造成洩漏或發送到一個未知位址或目的地。

4.當應用程式執行時，測試人員需注意來自外部不可信任資料來源輸入行為的風險。

5.Container View 另需注意行動應用程式非由使用者發起與其他應用程式或第三方服務通訊的風險。

安全開發生命週期：測試階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.5.1.防範惡意程式碼與避免資訊安全漏洞

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：NIST 800 SP-163 3.1.5,CSA MAST 4.2.2

參考來源：N/A

CPN-06:行動應用 App 如有包含直譯器程式碼作為輸入或是呼叫第三方 API，需謹慎處理運行中錯誤，避免遭惡意注入攻擊或版取得存取敏感資料權限。

說明：

執行期間解譯之程式碼可能使得不信任的惡意人士有機會造成一種解譯為未驗證碼輸入，例如：在遊戲的額外載入程序，腳本解譯的 SMS 表頭等。這提供惡意程式可以有機會規避 App 商店提供的安控機制。這可以導致注入攻擊並造成資料洩漏、監視、間諜軟體與撥號軟體(Diallerware)。此狀況並不是顯而易見，程式碼中可能包含一個解譯行為。惡意人士可尋找通過使用者輸入資料和使用第三方的 API 作為解譯使用者輸入訪問的任何功能，例如：JavaScript 解譯器。

安全原則：

- 最小化運行時解譯，並提供給運行時的解譯能力：以最低的特權等級運行解譯。
- 定義全面的轉譯語法為宜。
- 模糊測試解譯。
- 沙箱解譯。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.5.1.防範惡意程式碼與避免資訊安全漏洞

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG10.1, 10.2, 10.3,10.4

參考來源：N/A

CPN-07:行動應用 App 應避免使用內嵌瀏覽器元件技術和防止連線劫持。

說明：

iFrame 之應用可以將 Web/WAP 網站的資料嵌入 App 並進行資料互動。這種開發應用方式可能被包裝(Wrapper)網站執行點擊劫持攻擊。點擊劫持是利用熱門的網站服務，作為竊取資訊或將使用者重導向到攻擊者控制的網站的常見威脅。這類的 iFrame 攻擊主要目的是誘騙使用者點擊惡意程式，其目標是蒐集機密資訊或通過諸如跨網站程式碼(XSS Scripting)漏洞連結，取得受影響電腦的控制權。這種攻擊通常需要嵌入程式碼，這類超出一般使用者的知識範圍內所執行腳本的形式，可以在使用者不知情下點擊顯示執行或其他功能的按鈕來觸發。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

iOS：Objective-C

最好的方法是不使用內嵌瀏覽器元件。

- *謹慎使用 UIWebView : stringByEvaluatingJavaScriptFromString*

```
- (NSString *)stringByEvaluatingJavaScriptFromString:(NSString *)script
```

主機：Server

參考 Server-06。

行動應用 App 基本資安規範：4.1.5.1. 防範惡意程式碼與避免資訊安全漏洞

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：[iOS：stringByEvaluatingJavaScriptFromString - apple developer (查詢時間：2016/10/1)]

3.1.15. 行動應用程式完整性(O)

CPO-01:行動應用 App 程式碼應使用平台供應商核發企業或個人開發者憑證簽章。

說明：

為防止行動應用 App 被偽冒後被植入惡意程式，被上架至非 Android 或 iOS 官方平台商店供使用者下載，使用者無從分辨何者為安全 App，行動應用 App 程式碼應使用平台供應商核發企業或個人開發者憑證進行簽章。

Android：

開發者需使用有效的憑證來正確地簽章自己的 APKs。

注意：

一個正式上線的應用程式，需要被正式且有足夠有效期間的憑證來簽章。Google 建議至少要使用 2048 位元的版本。Keystore 內的正式憑證一定要被保護且只讓最少量的人可以存取。

下列指令可以產生一長度為 2048 位元的私鑰

```
$ keytool -genkey -v -keystore my-release-key.keystore -alias
alias_name -keyalg RSA -keysize 2048
```

iOS：

iOS 提供了多重保護來確保 App 經過簽署和驗證，且以沙箱(Sandbox)技術限制，進而保護使用者資料為了確保所有 App 均來自核准的已知來源且未被竄改，iOS 會要求所有可執行的程式碼均使用 Apple 核發的憑證進行簽署。裝置所隨附的 App(如「郵件」和 Safari)則由 Apple 簽署。第三方 App 也必須使用 Apple 核發的憑證進行驗證和簽署。強制性程式碼簽署將信任鏈的概念從作業系統延伸至 App，可防止第三方 App 載入未簽署的程式碼資源，或使用自行修改的程式碼。

若要在 iOS 裝置上開發並發布 App，開發者必須向 Apple 註冊並加入 Apple Developer Program(Apple 開發者計畫)。Apple 會先驗證每位開發者(無論是個人或企業)的真實身分，然後再核發憑證。開發者可使用該憑證對 App 進行簽署，並將其上傳至 App Store 進行發布。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.5.2.行動應用程式完整性

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：JSSEC AASDSCG .5.3

參考來源：

Android：Sign Your App（查詢時間：2016/10/1)

iOS：iOS Security Guide（查詢時間：2016/10/1) iOSApp 程式碼簽章(code sign)

CPO-02:行動應用 App 的程式碼儘量以原生方法撰寫，呼叫已載入記憶體的函式庫，以使惡意的程式碼分析難度提高。

說明：

可能在除錯或異常處理過程中創建不相關的功能或無意的機制，以下是與開發模糊處理相關的 3 個主要問題：

- **原生程式碼執行混淆**

  插入特定的功能到應用程式的原始程式碼，使得其執行過程中的原生程式碼將執行非預期的操作。

- **Call mapping 問題**

  Call Mapping 總是嵌入可以混淆控制流和摧毀一個行動應用程式的權限機制的功能。因此，任何呼叫映射相關功能均應被審核安全問題。

- **重建性混淆**

  通過拆卸的行動應用程式的原始程式碼部分及重新組裝測試後的程式碼，正常沙箱保護將不能檢測到相關的風險。開發人員必須了解使用混淆程式碼的目的或測試混淆程式碼的結果，而審查行動應用程式。

開發實作人員使用原生方法呼叫通常已經被載入到記憶體中的函式庫。這些方法提供了一個應用程式重用寫在不同的語言程式碼的方法。這些呼叫可以提供某種程度模糊影響執行應用程式的分析的能力。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.5.2.行動應用程式完整性

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：NIST 800 SP-163 3.1.5.,CSA MAST 4.2.5

參考來源：N/A

CPO-03:行動應用 App 程式碼應運用人工或工具使之增加複雜度，並輔以限制除錯器使用、反追蹤、二進位剝離等措施，使惡意人士使用逆向工程方法分析程式碼難度增加。

說明：

由於 App 安裝到手機端之後，就會暴露在使用者或攻擊者手中。

藉由使用軟體逆向工程工具，攻擊者可以逆向解開應用程式程式碼，並窺視程式的邏輯、破解程式保護，甚或盜取程式碼；另一方面，當手機端 App 被破解也會使得 Server 端的服務被窺伺或破解。為了防止此問題，應適當加入保護措施，以保護程式碼內容與 Server 主機安全。

保護措施的建議方向有下列 5 項，限於時程及成本，至少實作第 1 項或第 2 項，其餘視需要增加：

1.增加程式碼複雜度

將 Android App(.apk 文件)進行逆向工程相當容易達成，然後可以檢查 App 的內部工作。建議針對程式碼進行模糊處理，以增加惡意攻擊者窺視程式的邏輯、破解程式保護或盜取程式碼的困難度。

iOS 應用程式由於它們的設計方式，也容易被逆向工程的攻擊。一個應用程式的類(classes)和協議(protocols)都儲存在 object file 中，允許攻擊者完全映射出 App 的設計。

Objective-C 的本身是一種反射的語言，能夠感知和修改自己的狀態。使用合適的工具的攻擊者可以用相同的方式感知和修改所述 runtime 管理的 App 狀態。

Objective-C 包含一個簡單的訊息傳遞框架，這是非常容易追蹤，並且可以被操縱以攔截或甚至於 App 的 runtime 時篡改。在 Objective-C runtime 時執行繞過認證和政策檢查的攻擊是相對簡單的，而內部應用程式的完整性檢查或邏輯檢查應符合應用程式開發的管理要求。

2.使用程式碼保護工具：

如果應用程式處理高度敏感的資料，考慮實施反除錯技術。各種技術併用可以增加逆向工程反組繹程式碼的複雜性。一種方法是使用 C/C ++限制攻擊者輕易於 runtime 時控制。有充足的 C 和 C ++程式庫是非常成熟且易於使用的 Objective-C 整合和 Android 提供了 JNI。在 iOS 考慮在底層編寫程式碼密鑰部分由 Objective-C 避免於 Run-time 時暴露和被操控。可以 Objective-C 的逆向工程工具，如 class-dump，class-dump-z 和 Cycript 進行測試。

3.限制除錯器

通過整合的系統呼叫，應用程式可以指定作業系統不應允許一個除錯器附加到處理程序(Process)。通過防止除錯附著，攻擊者的能力對於底層運行時干擾是有限的。攻擊者為了攻擊必須先繞過在底層的應用程式除錯限制。這進一步增加了攻擊的複雜性。Android App 應該在應用程式清單，以防止易於運行時操作被攻擊或惡意軟體 android:debuggable =“false”的設定。

4.追蹤檢查

應用程式可以確定它是否是當前正由除錯器或其他除錯工具追蹤。如果被追蹤，應用程式可以執行任何數量的可能的攻擊的反應動作，如丟棄加密密鑰來保護使用者資料，試圖為自己辯護，通知伺服器管理員或者其他這種類型的反應。

5.優化、以降低可視度

要隱藏高階數學計算和其他類型的複雜的邏輯，利用編譯程序優化操作可以幫助混淆對象程式碼，以便它不能容易地被攻擊者逆向還原，使其難以使得攻擊者對於特殊程式碼的理解。在 Android 中可以容易通過利用與本地 NDK 編譯函式庫來達成需求。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

Android：Java

使用 Google 提供的 ProGuard，來將程式碼進行混淆。

在 bulid.gradle 檔案中，把 minifyEnabled 改成 true 即可啟動

```
buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-
android.txt'), 'proguard-rules.pro'
        }
    }
```

iOS：Objective-C

提供偵測裝置是否處於 jail-broken 模式原始碼

```
+(BOOL)isJailbroken{
   #if !(TARGET_IPHONE_SIMULATOR)
   if ([[NSFileManager defaultManager] fileExistsAtPath:@"/Applications/Cydia.app"]){
       return YES;
   }else if([[NSFileManager defaultManager]
fileExistsAtPath:@"/Library/MobileSubstrate/MobileSubstrate.dylib"]){
       return YES;
   }else if([[NSFileManager defaultManager] fileExistsAtPath:@"/bin/bash"]){
       return YES;
   }else if([[NSFileManager defaultManager] fileExistsAtPath:@"/usr/sbin/sshd"]){
       return YES;
   }else if([[NSFileManager defaultManager] fileExistsAtPath:@"/etc/apt"]){
       return YES;
   }
```

```
    if([[UIApplication sharedApplication] canOpenURL:[NSURL
URLWithString:@"cydia://package/com.example.package"]]){
        //Device is jailbroken
        return YES;
    }
    //try write /private
    NSError *error;
    NSString *stringToBeWritten = @"This is a test.";
    [stringToBeWritten writeToFile:@"/private/jailbreak.txt" atomically:YES
encoding:NSUTF8StringEncoding error:&error];
    if(error==nil){
        //Device is jailbroken
        return YES;
    } else {
        [[NSFileManager defaultManager] removeItemAtPath:@"/private/jailbreak.txt" error:nil];
    }
    #endif
    //All checks have failed. Most probably, the device is not jailbroken
    return NO;
}
```

iOS:提供拒絕 DEBUG 模式原始碼(for release version)

```
//拒絕 DEBUG 模式的執行
#import <dlfcn.h>
#import <sys/types.h>
typedef int (*ptrace_ptr_t)(int _request, pid_t _pid, caddr_t _addr, int _data);
#if !defined(PT_DENY_ATTACH)
#define PT_DENY_ATTACH 31
#endif
void disable_gdb() {
    void* handle = dlopen(0, RTLD_GLOBAL | RTLD_NOW);
    ptrace_ptr_t ptrace_ptr = dlsym(handle, "ptrace");
    ptrace_ptr(PT_DENY_ATTACH, 0, 0, 0);
    dlclose(handle);
}
int main(int argc, char * argv[]) {
    #if !(DEBUG) // Don't interfere with Xcode debugging sessions.
        disable_gdb();
    #endif
```

```
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil, NSStringFromClass([AppDelegate class]));
    }
}
```

iOS 混亂器(obfuscator)說明

iOS 系統內定安裝後，已經實作了Class rename；而字串的保護需自行實作；而降低可視度，通常也實現在混亂器工具中

例如：PreEmptive Protection for iOS 實現如下的內容

降低可視度

```
int max(int x, int y) {
    int switchvar = 1942843;
    while (true) {
        switch (switchvar) {
            case 598232:
                return y;
            case 1942843:
                if (x >= y) {
                    switchvar = 8289314;
                } else {
                    switchvar = 598232;
                }
              break;
            case 8289314:
                return x;
        }
    }
}
```

行動應用 App 基本資安規範：4.1.5.2.行動應用程式完整性

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：

Android：

http://developer.android.com/guide/publishing/licensing.html - app-obfuscation(查詢時間：2016/10/1)

Android-ProGuard:http://proguard.sourceforge.net/ -(查詢時間：2016/10/1)

http://developer.android.com/tools/help/proguard.html(查詢時間：2016/10/1)

Android-DexGuard:http://www.saikoa.com/dexguard(查詢時間：2016/10/1)

iOS：

Bugging Debuggers - disable debug(查詢時間：2016/10/1)

iOS-application jailbreak-detection(查詢時間：2016/10/1)

CWE/OWASP：

M8 - Security Decisions via Untrusted Inputs(查詢時間：2016/10/1)

M10 - Lack of Binary Protections(查詢時間：2016/10/1)

CWE 656(查詢時間：2016/10/1)

CPO-04:行動應用 App 應程式碼避免使用過於簡單安全設計邏輯，可增加如挑戰/回應、OTP、定時回調(callback)、基於密碼學驗證等增加對攻擊者分析程式碼難度。

說明：

在程式碼中簡單的邏輯測試更加容易受到攻擊。例：if sessionIsTrusted == 1，這是一個簡單的邏輯檢查，如果攻擊者可以改變一個值，它們可以規避安全控制。這些邏輯測試是容易規避在許多層面上。在組合語言層級，攻擊者可以僅透過一個除錯器來找到合適 iOS 應用程式的攻擊點，以 CBZ(compare-and-branch-on-zero) 或 CBNZ(compare-and-branch- on-nonzero)指令進行攻擊，並進行逆向工程。這可以在運行時通過簡單地掃描攻擊對象的記憶體地址和在應用程式運行中改變其實例變量。在 Android 上，應用程式可以被反編譯為 SMALI 並作為與重新編譯前的分支條件修補。

為了加強保護能力，建議程式加強判斷的複雜度，或改變變數名稱，以保護程式。另一方面提醒：手機的功能與資料流的控制，須由主機端控制；而非由手機端，單方面邏輯控制功能與資料之權限。

此外可增加如挑戰/回應、OTP、定時回調(callback)、基於密碼學驗證等增加對攻擊者分析程式碼難度。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.5.2.行動應用程式完整性

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：CWE/OWASP：

M8 - Security Decisions via Untrusted Inputs(查詢時間：2016/10/1)
M10 - Lack of Binary Protections(查詢時間：2016/10/1)
CWE 200(查詢時間：2016/10/1)

CPO-05:行動應用 App 為保障程式碼及敏感性資料完整性，應實作反篡改技術，如程式碼簽章及資料完整性校驗(Checksum)。

說明：

應用程式可能被攻擊者篡改或設置後門並重新簽署發布，因此程式需要使用防止篡改技術，以及防止執行階段與其他程序的干擾與篡改。建議使用檢查碼、數位簽章和其他驗證機制，可以防止此情形。而當發生此情形時，應用程式除了可以先抹除或保護使用者重要資料，也可警告使用者，或通知系統管理員。

附上檢查破解的程式碼。附上拒絕 DEBUG 的程式碼。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

Android：Java

本範例提供一 TamperCheck Class 可以使用此程式來判斷是否此 APP 仍是由當時開發人員簽章所簽証的

```
import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.content.pm.Signature;

public class TamperCheck {

//we store the hash of the signture for a little more protection
private static final String APP_SIGNATURE = "1038C0E34658923C4192E61B16846";

  /**
     * Query the signature for this application to detect whether it matches the
     * signature of the real developer. If it doesn't theAppmust have been
     * resigned, which indicates it may been tampered with.
     *
     * @param context
     * @return true if the app's signature matches the expected signature.
     * @throws NameNotFoundException
     */
    public boolean validateAppSignature(Context context) throws NameNotFoundException
{

        PackageInfo packageInfo = context.getPackageManager().getPackageInfo(
```

```
              getPackageName(), PackageManager.GET_SIGNATURES);
    //note sample just checks the first signature
        for (Signature signature : packageInfo.signatures) {
            // SHA1 the signature
            String sha1 = getSHA1(signature.toByteArray());
            // check is matches hardcoded value
            return APP_SIGNATURE.equals(sha1);
        }

        return false;
    }

  //computed the sha1 hash of the signature
  public static String getSHA1(byte[] sig) {
        MessageDigest digest = MessageDigest.getInstance("SHA1", "BC");
            digest.update(sig);
            byte[] hashtext = digest.digest();
            return bytesToHex(hashtext);
    }

  //util method to convert byte array to hex string
  public static String bytesToHex(byte[] bytes) {
    final char[] hexArray = { '0', '1', '2', '3', '4', '5', '6', '7', '8',
                '9', 'A', 'B', 'C', 'D', 'E', 'F' };
        char[] hexChars = new char[bytes.length * 2];
        int v;
        for (int j = 0; j < bytes.length; j++) {
            v = bytes[j] & 0xFF;
            hexChars[j * 2] = hexArray[v >>> 4];
            hexChars[j * 2 + 1] = hexArray[v & 0x0F];
        }
        return new String(hexChars);
    }
  }
```

iOS：參考 CPO-03 安全程式碼

行動應用 App 基本資安規範：4.1.5.2. 行動應用程式完整性

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：

Android：https://gist.github.com/scottyab/b849701972d57cf9562e(查詢時間：2016/10/1)

iOS：反破解偵測程式碼

http://stackoverflow.com/questions/6530364/how-to-detect-that-the-app-is-running-on-a-jailbroken-device(查詢時間：2016/10/1)

CWE/OWASP：

M10 - Lack of Binary Protections(查詢時間：2016/10/1)

CWE 354(查詢時間：2016/10/1)

3.1.16. 函式庫引用安全(P)

CPP-01:行動應用 App 使用第三方函式庫前，需先確認其是來自可靠來源、有持續更新並經測試沒有漏洞、後端木馬及不明傳送目的地。

說明：

行動應用 App 當引用第三方函式庫時需了解其來源可靠性，務必更新到最新版本的並進行使用前測試，需注意漏洞及檢測是否有不明伺服器端連線，以避免安全性問題。若自行編譯函式庫時，也需注意元件組成是否安全，是否有惡意程式碼被夾帶進來。著名的 XCodeGhost 事件，或者更多的駭客、病毒的注入，都是類似的作法。包含 Android 及 iOS 的編譯紀錄均應詳細檢查。在 Unix-like 的作業系統，需注意 LD_PRELOAD 的使用，並可透過 set-x 的方式，檢查編譯紀錄是否有惡意的注入。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.5.3.函式庫引用安全

行動應用 App 基本資安檢測基準

4.1.5.3.1. 檢查行動應用程式引用之函式庫是否不存在已知安全性漏洞。

各國開發要點參考：ENISA SSDG6.1, NIST 800 SP-163 3.1.5.b, CSA MAST 4.2.3

參考來源：CWE/OWASP：

M8 - Security Decisions via Untrusted Inputs (查詢時間：2016/10/1)

CWE 829 (查詢時間：2016/10/1)

CPP-02:行動應用 App 使用第三方 SDK/API，需先確認其是來自可靠來源、有持續更新並經測試沒有漏洞、後端木馬及不明傳送目的地。

說明：

開發人員可能使用到黑名單或惡意套件來開發行動應用程式出現惡意行為或過失。作為套件其安全性可能尚未通過測試，如果 API 呼叫方法具有缺陷，將使應用程式容易受到攻擊導致資訊洩露後果。這些惡意套件存在於行動應用 App 風險應不允許發生。行動應用 App 當引用第三方 SDK/API 時需了解其來源可靠性，務必更新到最新版本的並進行使用前測試，需注意漏洞及檢測是否有不明伺服器端連線，以避免安全性問題。

為了規避有風險的第三方 SDK/API，組織可以設定政策只使用軟體供應商原生軟體開發工具(SDK)提供 API 並適當使用。其他的 API 不可被允許使用，而對准許使用 API 應注意其資料傳輸的流動與目的地。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.5.3.函式庫引用安全

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 6.2, NIST 800 SP-163 3.1.3.d, CSA MAST4.2.3

參考來源：N/A

3.1.17. 使用者輸入驗證(Q)

CPQ-01:行動應用 App 應實作驗證使用者輸入字串資料型別及長度之正確性，避免惡意輸入導致應用程式毀損、緩衝區溢位、各種注入攻擊發生。

說明：

即使是從應用程式產生的資料也有可能被截取和處理，這可能導致包括應用程式意外中止(產生含密鑰的日誌)攻擊，緩衝區溢出，SQL INJECTION 等攻擊。在 iOS 中，這可以很容易地透過實作 UITextFieldDelegate 的方式進行防範。

實作適當的 Web 應用程式輸入安全控制機制，預設從客戶端的所有輸入應必須被視為不可信，而必須被有條件的驗證與處理。服務必須透過過濾和從應用程式和使用者驗證輸入，適當有條件的阻擋惡意輸入，輸入包括傳送前和所有的使用者輸入接收。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

iOS: Objective-C

驗證輸入長度、格式Email

如

```
- (BOOL) validEmail:(NSString*) emailString {
    if([emailString length]==0){
        return NO;
    }
    NSString *regExPattern = @"[A-Z0-9a-z._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,4}";
    NSRegularExpression *regEx = [[NSRegularExpression alloc]
initWithPattern:regExPattern options:NSRegularExpressionCaseInsensitive error:nil];
    NSUInteger regExMatches = [regEx numberOfMatchesInString:emailString options:0
range:NSMakeRange(0, [emailString length])];
    if (regExMatches == 0) {
        return NO;
    } else {
        return YES;
    }
}
```

網頁主機： Server .NET C#

- *驗證輸入長度、格式*

  .NET 使用 RegularExpressionValidator ，使用正規表示式來驗證使用者輸入的格式。

```
例如：檢查英數字輸入 6 至 10 位。
ValidationExpression="[a-zA-Z]{6,10}"
```

行動應用 App 基本資安規範： 4.1.5.4.使用者輸入驗證

行動應用 App 基本資安檢測基準

4.1.5.4.1.(1)檢查行動應用程式是否針對使用者輸入字串驗證型別。

4.1.5.4.1.(2)檢查行動應用程式是否針對使用者輸入字串驗證長度。

各國開發要點參考：JSSEC AASDSCG 3.2

參考來源：

iOS：NSRegularExpression Class - iOS develpoer(查詢時間：2016/10/1)

網站主機：User Input Validation in ASP.NET(查詢時間：2016/10/1)

CPQ-02:行動應用 App 需要驗證使用者輸入、伺服器端傳入及其他裝置輸入之資料，防止因 Buffer overflow/underflow 造成安全性漏洞。

說明：

因為程式設計缺陷造成緩衝區溢位或不足位，在 C、Objective-C 與 C++程式碼中的堆疊(Stack)和堆積(heap)是安全漏洞的主要來源。雖然現今主流作業系統都有 buffer overflow protection(緩衝區溢位保護/記憶體位置重新導向)，且多數程式語言都會檢查輸入防止 buffer overflow 或 underflow，但是 C、Objective-C 的，和 C++則無此功能。

建議可以遵循以下規則，讓此種攻擊的發生機率下降：

- 在需要使用緩衝區前，以"0"填滿所有的緩衝區；讓緩衝區的內容僅包含"0"，沒有任何敏感資訊。
- 經常檢查回傳值和失敗回傳。
- 當呼叫配置(allocation)或初始化(initialization)的函數(例如：AuthorizationCopyRights)失敗時，不要使用其結果值，因為它有可能是舊的資料。
- 你可以使用「read」及其他類似的系統呼叫(System call)，以確認實際上所讀取的資料量，而不是使用一個預設的常數(constant)。如果執行系統呼叫後的回傳值不是預期的資料數量，就需視為執行失敗(fail)。
- 當程式在執行一個有字元數量限制的資料結構(data structure)時，務必驗證所輸入的字元是否如為預期。
- 避免將非 C 字串(CFStringRef 物件、NSString 物件、CFDataRef 物件、帕斯卡串等)轉換成 C 字串，儘量直接使用既有的格式就好。如果真的要轉換，請務必檢查轉換後的 C 字串的長度，或是檢查其中有無空字元(Null)。
- 避免混合緩衝操作和字串操作，如果真的有必要使用，務必檢查轉換後的 C 字串的長度，或是檢查其中有無空字元(Null)。
- 儲存檔案時應以得防止被惡意篡改或截斷的方式進行。
- 避免整數值(Integer)的 overflow 和 underflow。

安全開發生命週期：開發實作階段

不安全程式碼範例：

iOS：Objective-C

- *strcat,strcpy,strncat,strncpy,sprintf,vsprintf,gets*

函式不會控制資料結尾，除了長度不安全以外，也會造成沒有結尾。

```
int len = (int)self.myInput.text.length;
    char buf[len+1];
    //strncpy
    strncpy(buf,  [self.myInput.text UTF8String] , len);
```

安全程式碼範例：

iOS：Objective-C

- *strlcat,strlcpy,strlcat,strlcpy,snprintf,asprintf,vsnprintf,vasprintf,fgets*

函式會控制資料結尾，除了長度安全以外，也會有結尾。

```
int len = (int)self.myInput.text.length;
    char buf[len+1];
    //strlpy
    strlcpy(buf, [self.myInput.text UTF8String] , len);
```

行動應用 App 基本資安規範：4.1.5.4.使用者輸入驗證

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：JSSEC AASDSCG 3.2

參考來源：

緩衝區溢位,Wiki (查詢時間：2016/10/1)

SecureCodingGuide - BufferOverflows - iOS developer (查詢時間：2016/10/1)

CPQ-03:行動應用 App 提供使用者輸入值儘量可以參數化(Query parameterization)。

說明：

由於 SQL 命令，若欄位參數採用字串組合，遇到特殊字元或惡意攻擊，會有安全漏洞，必須改用元件提供之欄位參數輸入方式。如下圖程式碼範例所示，如何建構最常見的程式語言的參數化查詢。這些程式碼的目的是展示給 Web 開發人員如何避免在 Web 應用程式中建立資料庫查詢的時候遭受 SQL Injection 攻擊。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：

Language - Library	Parameterized Query
Java - Standard	• String custname = request.getParameter("customerName"); • String query = "SELECT account_balance FROM user_data WHERE user_name = ? "; • **PreparedStatement pstmt = connection.prepareStatement(query);** • **pstmt.setString(1, custname);** • ResultSet results = pstmt.executeQuery( );

Java - Hibernate	• //HQL
	• Query safeHQLQuery = session.createQuery("from Inventory where productID=:productid");
	• safeHQLQuery.setParameter("productid", userSuppliedParameter);
	• //Criteria API
	• String userSuppliedParameter = request.getParameter("Product-Description"); // This should REALLY be validated too
	• // perform input validation to detect attacks
	• Inventory inv = (Inventory) session.createCriteria(Inventory.class).add
	• (Restrictions.eq("productDescription", userSuppliedParameter)).uniqueResult();

Language - Library	Parameterized Query

.NET/C#	• String query = "SELECT account_balance FROM user_data WHERE user_name = ?";
	• try {
	• OleDbCommand command = new OleDbCommand(query, connection);
	• **command.Parameters.Add(new OleDbParameter("customerName", CustomerName Name.Text));**
	• OleDbDataReader reader = command.ExecuteReader();
	• // …
	• } catch (OleDbException se) {
	• // error handling
	• }

ASP.NET	• string sql = "SELECT * FROM Customers WHERE CustomerId = @CustomerId"; • **SqlCommand command = new SqlCommand(sql);** • **command.Parameters.Add(new SqlParameter("@CustomerId", System.Data.SqlDbType.Int));** • command.Parameters["@CustomerId"].Value = 1;
Ruby - ActiveRecord	• **# Create** • Project.create!(:name => 'owasp') • **# Read** • Project.all(:conditions => "name = ?", name) • Project.all(:conditions => { :name => name }) • Project.where("name = :name", :name => name) • **# Update** • project.update_attributes(:name => 'owasp') • **# Delete** • Project.delete(:name => 'name')

Language - Library	Parameterized Query
Ruby	• insert_new_user = db.prepare "INSERT INTO users (name, age, gender) VALUES (?, ? ,?)" • insert_new_user.execute 'aizatto', '20', 'male'
PHP - PDO	• $stmt = $dbh->prepare("INSERT INTO REGISTRY (name, value) VALUES (:name, :value)"); • **$stmt->bindParam(':name', $name);** • **$stmt->bindParam(':value', $value);**
Cold Fusion	• <cfquery name = "getFirst" dataSource = "cfsnippets"> • **SELECT * FROM #strDatabasePrefix#_courses WHERE intCourseID =** • **<cfqueryparam value = #intCourseID# CFSQLType = "CF_SQL_INTEGER">** • </cfquery>
Perl - DBI	• my $sql = "INSERT INTO foo (bar, baz) VALUES (?, ?)"; • **my $sth = $dbh->prepare($sql);** • **$sth->execute($bar, $baz);**

行動應用 App 基本資安規範：4.1.5.4.使用者輸入驗證

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：[Query_Parameterization_Cheat_Sheet - owasp.org (查詢時間：2016/10/1)]

CPQ-04:行動應用 App 需要驗證使用者輸入及伺服器端傳入資料，防止因 Integer overflow and underflow 造成安全性漏洞。

說明：

整數溢位或不足位漏洞會導致非預期結果，目前主流 CPU 使用 32 位元或 64 位元暫存器來表示整數的大小有限，當算術運算產生的結果超出有限的表示範圍，超出的位元(Overflow)將被捨棄，未超出位元將儲存在暫存器中。

Android 在 5.1.1 版以前的版本有此漏洞，5.1.1 以後版本已修正此漏洞。

iOS 的 Objective-c 仍不會自動驗證，需要以程式實作控制。

安全開發生命週期：開發實作階段

不安全程式碼範例：

iOS：Objective-C

iOS 因造成整數溢位(Object C)

```
[NSString stringWithFormat:@"INT32_MAX=%d", INT32_MAX];
[NSString stringWithFormat:@"INT32_MAX+1=%d", INT32_MAX+1];
[NSString stringWithFormat:@"INT32_MAX+1+INT32_MAX=%d", INT32_MAX+1+INT32_MAX];
[NSString stringWithFormat:@"INT32_MIN=%d", INT32_MIN];
[NSString stringWithFormat:@"INT32_MIN-1=%d", INT32_MIN-1];
[NSString stringWithFormat:@"INT32_MIN-1-INT32_MIN=%d", INT32_MIN-1-INT32_MIN];
```

造成整數溢位且 Exception(Swift)

```
var potentialOverflow = Int16.max
// potentialOverflow equals 32767, which is the maximum value an Int16 can hold
potentialOverflow += 1
```

造成整數溢位(Swift)Overflow 運算子

```
var unsignedOverflow = UInt8.max
// unsignedOverflow equals 255, which is the maximum value a UInt8 can hold
unsignedOverflow = unsignedOverflow &+ 1
// unsignedOverflow is now equal to 0
```

安全程式碼範例：

iOS：Objective-C

由於整數會有溢位問題，在加減乘的運算時，都會發生。請務必注意。除了採用安全的計算方法或限制以外。在計算記憶體空間大小時，也必須檢查是否計算出來的數字大小，是否會超過 SIZE_MAX。

```
//calculate n*m with overflow-safe 計算記憶體空間大小
if (n > 0 && m > 0 && SIZE_MAX/n >= m) {
   size_t bytes = n * m;
   ... /* allocate “bytes” space */
}
```

行動應用 App 基本資安規範：4.1.5.4.使用者輸入驗證

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：JSSEC AASDSCG 3.2

參考來源：iOS

- SecureCodingGuide - BufferOverflows - iOS developer (查詢時間：2016/10/1)
- 計算緩衝區大小 (查詢時間：2016/10/1)
- Swift Programming Language/AdvancedOperators.html - iOS developer (查詢時間：2016/10/1)

CPQ-05:行動應用 App 應實作過濾使用者輸入及伺服器端傳入資料中易導致 SQL Injection 之字串。

說明：

由於 SQL 命令，若欄位參數採用字串組合，遇到特殊字元或惡意攻擊，會有安全漏洞，必須改用元件提供之欄位參數輸入方式。

於 CPQ-03 也說明相關解決方式：使用 Query parameterization。

安全開發生命週期：開發實作階段

不安全程式碼範例：

Android：Java

SQL 語法直接使用+的法式將使用者的輸入串連起來

```
tring commandString = "INSERT INTO mytable (idno, name) VALUES
'"+userInputIdno+","+userInputName+"'";
try {
    mSampleDB.execSQL(commandString);
}catch (SQLException e) {
    ......
} finally {
    sqlStmt.close();
}
```

iOS：Objective-C

sqlite 錯誤方式

```
- (IBAction) insertUnsafe : (id) sender
{
    BOOL rst = [self executeSQL:[NSString stringWithFormat:@"INSERT INTO
Setting(SKey,SValue) VALUES('%@','%@')", self.edit_key.text, self.edit_value.text]];
}
```

安全程式碼範例：

Android：Java

使用參數化方式避免使用者的惡意輸入

```
String commandString = "INSERT INTO mytable (idno, name) VALUES (?, ?)";
//使用參數化的SQLiteStatement以防止SQL Injection
SQLiteStatement sqlStmt = mSampleDB.compileStatement(commandString);
sqlStmt.bindString(1, userInputIdno);
sqlStmt.bindString(2, userInputName);
try {
    sqlStmt.executeInsert();
```

```
}catch (SQLException e) {
    ......
} finally {
    sqlStmt.close();
}
```

iOS：Objective-C

資料庫：sqlite 正確方式，即CPQ-03：Query parameterization

```
- (IBAction) insertSafe : (id) sender
{
    sqlite3_stmt *statement;
    BOOL DONE = TRUE;
    char *sql = "INSERT INTO Setting(SKey,SValue) VALUES(?,?)";

    if (sqlite3_prepare_v2(database,sql,-1,&statement,NULL) == SQLITE_OK) {
        sqlite3_bind_text(statement,1,[self.edit_key.text UTF8String],-1,
SQLITE_TRANSIENT);
        sqlite3_bind_text(statement,2,[self.edit_value.text UTF8String],-1,
SQLITE_TRANSIENT);
        if (sqlite3_step(statement) != SQLITE_DONE) {
            DONE = FALSE;
            NSLog(@"Error");
        }
    }
    else
        DONE = FALSE;
    sqlite3_finalize(statement);
}
```

行動應用 App 基本資安規範：4.1.5.4.使用者輸入驗證

行動應用 App 基本資安檢測基準：

4.1.5.4.2.(1) 檢查行動應用程式是否過濾導致 SQL Injection 之字串。

各國開發要點參考：CSA MAST 4.2.3, JSSEC AASDSCG 3.2

參考來源：iOS:SecureCodingGuide - injections - iOS developer(查詢時間：2016/10/1)

CPQ-06:行動應用 App 應實作過濾使用者輸入及伺服器端傳入資料中易導致 Command injection 之字串。

說明：

命令注入是一種攻擊，其目標是透過一個易受攻擊的應用程式以間接執行作業系統上的任意命令。當應用程式通過攻擊者提供的惡意資料(表單，Cookie，HTTP 表頭等)，而進入到系統 shell 命令注入攻擊。在這種攻擊中，攻擊者操作的系統命令通常與產生漏洞應用程式的權限執行相關聯。命令注入攻擊主要產生原因多是由於輸入驗證不足。

例如：Linux 處理 Escape 分號;字元造成命令的執行

輸入 Story.txt;more /etc/passwd 造成資料的洩漏。

例如：Linux 環境變數可能被改變，而使得被換成其他的不預期的執行檔

```
char* home=getenv("APPHOME");
char* cmd=(char*)malloc(strlen(home)+strlen(INITCMD));
strcpy(cmd,home);
strcat(cmd, INITCMD);
execl(cmd, NULL);
```

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

網站主機：Server .NET C#

檢查合法檔名

```
var isValid = !string.IsNullOrEmpty(fileName) &&
fileName.IndexOfAny(Path.GetInvalidFileNameChars());
```

行動應用 App 基本資安規範：4.1.5.4.使用者輸入驗證

行動應用 App 基本資安檢測基準：

4.1.5.4.2.(3) 檢查行動應用程式是否過濾導致 Command Injection 之字串。

各國開發要點參考：CSA MAST 4.2.3 , JSSEC AASDSCG 3.2

參考來源：

- iOS：SecureCodingGuide - injections - iOS developer (查詢時間：2016/10/1)
- 網站主機：.path.getinvalidfilenamechars - .net developer (查詢時間：2016/10/1)

CPQ-07:行動應用 App 應避免讓使用者傳入文件傳入檔案系統或框架 API，如為必須允許，實作過濾使用者輸入及伺服器端傳入資料中易導致 Local File Inclusion 之字串。

說明：

文件包含漏洞是一個容易在網站上發現的漏洞，允許攻擊者產製一個文件通過對一個具有漏洞腳本的 Web 伺服器進行攻擊。該漏洞發生由於惡意使用者提供未經適當驗證的輸入。這可能會導致作為最小為輸出文件或更嚴重的事件，如以下內容：

- 在代碼執行客戶端這樣的 JavaScript 這可能會導致諸如其他攻擊跨網站程式碼(XSS)
- 拒絕服務(DOS)
- 數據失竊 /操作

最好的方法為避免讓使用者傳入文件傳入檔案系統或框架 API，如果必要允許前述行為必須過濾是否有包含可執行 Local File Inclusion 之字串，實作可參考 https://www.owasp.org/index.php/Testing_for_Local_File_Inclusion(查詢時間：2016/10/1)

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.5.4.使用者輸入驗證

行動應用 App 基本資安檢測基準：

4.1.5.4.2.(4) 檢查行動應用程式是否過濾導致 Local File Inclusion 之字串。

各國開發要點參考：JSSEC AASDSCG 3.2

參考來源：

- https://www.owasp.org/index.php/Testing_for_Local_File_Inclusion (查詢時間：2016/10/1)
- https://en.wikipedia.org/wiki/File_inclusion_vulnerability (查詢時間：2016/10/1)

CPQ-08:行動應用 App 應實作過濾使用者輸入及伺服器端傳入資料中易導致 XML Injection 之字串。

說明：

XML 注入攻擊者會嘗試在 SOAP 訊息中插入各種字串，目標在對 XML 結構注入各種 XML 標籤。通常一個成功的 XML 注入攻擊將導致限制操作的執行。根據各種執行的操作，而衍生各種安全問題。

由於 XML 的應用，在存取 XML 資料的時候，也須避免如 SQL inject 的處理方式而造成的漏洞。或基於 XML 的衍生功能與漏洞進行限制，例如：XML External Entity (XXE)。

安全開發生命週期：開發實作階段

不安全程式碼範例：

網頁主機：Server .NET C#

使用字串加減維護 XML

```
xmlstr = "<username>" + "foo<" + "</username>";
```

安全程式碼範例：

Android；Java

```
libxml_disable_entity_loader(true);
```

網頁主機：Server .NET C#

使用元件維護 XML

```
XmlNode node = doc.CreateNode("username");
node.Value = "foo<";
```

行動應用 App 基本資安規範：4.1.5.4.使用者輸入驗證

行動應用 App 基本資安檢測基準：

4.1.5.4.2.(5)檢查行動應用程式是否過濾導致 XML Injection 之字串。

各國開發要點參考：N/A

參考來源：

iOS：SecureCodingGuide - injections - iOS developer (查詢時間：2016/10/1)

網頁主機：system.xml.xmlnode.value (查詢時間：2016/10/1)

CPQ-09:行動應用 App 應實作過濾使用者輸入及伺服器端傳入資料中易導致 Format String Injection 之字串。

說明：

從使用者或其他信任的來源進行輸入和顯示出來，顯示的函式有可能造成問題而需小心應對。

安全開發生命週期：開發實作階段

不安全程式碼範例：iOS：Objective-C

```
printf ("%08x %08x %08x %08x %08x\n");
//NSString *bug = @"%@%@";
NSLog(bug);
```

安全程式碼範例：iOS：Objective-C

```
//NSString *bug = @"%@%@";
NSLog(“bug is %@”,bug);
```

網站主機：Server Linux

```
syslog(LOG_INFO, "%s", cmdBuf);
```

行動應用 App 基本資安規範：4.1.5.4.使用者輸入驗證

行動應用 App 基本資安檢測基準：

4.1.5.4.2.(6)檢查行動應用程式是否過濾導致 Format String Injection 之字串。

各國開發要點參考：N/A

參考來源：SecureCodingGuide - ValidatingInput - iOS developer (查詢時間：2016/10/1)

CPQ-10:確認行動應用 App 授權功能使用者介面，可正常如預期運作，不論顯示幕大小、橫向或直向，如允許輸入，需可帶出虛擬鍵盤，其顯示與按鍵需能如預期。

說明：

虛擬鍵盤帶出在觸控螢幕位置如果不精確，會導致使用者輸入非預期字元，需要反覆測試使用以手指按壓，於觸控螢幕位置上的虛擬鍵盤位置與輸入至行動應用 App 是否一致。

安全開發生命週期：開發實作、測試階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.5.4.使用者輸入驗證

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：NIST SP 800-163 3.1.1

參考來源：N/A

3.2.Android 安全開發實務

Android-01:Android 版本行動應用 App 應謹慎實作 Intents，避免資訊不慎外洩遭惡意運用。

說明：

意圖元件(Intents)是使用在內部元件信號傳遞及以下功能:

1.開啟另一個使用者界面，啟動另一個 Activity。

2.作為廣播事件，以通知系統和應用程式特定狀態的改變。

3.啟動、停止背景程式及與其溝通。

4.經由 ContenetProviders 來存取資料。

5.扮演回調(Callbacks)來處理相關連的事件。

不適當的實作有可能會造成資料洩密、限制使用的功能被不當使用與程式執行流程被調整或繞過。

安全開發生命週期：開發實作階段

不安全程式碼範例： Java

使用不指定接收 Class 的 Intent 來傳遞機密資訊，是不安全的也可能會被攔截，而使得資料外洩。

```
Intent intent = new Intent();
intent.putExtra("RESULT", "Sensitive Info");
startActivity(intent);
```

安全程式碼範例： Java

當使用 Intent 來傳遞資料給 Activity, 為維持資料安全，請使用明文指定接收 Class 的名稱。

```
//明文指定接收此 intent 的類別，以避免其他程式攔截取得資訊，或改變程式流程
Intent intent = new Intent(this, LoginActivity.class);
intent.putExtra("RESULT", "Not Sensitive Info");
startActivity(intent);
```

行動應用 App 基本資安規範：4.1.5.1.防範惡意程式碼與避免資訊安全漏洞

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：CWE/OWASP：

M8 - Security Decisions via Untrusted Inputs(查詢時間：2016/10/1)

M10 - Lack of Binary Protections(查詢時間：2016/10/1)

CWE 927(查詢時間：2016/10/1)

Android-02:Android 版行動應用 App 應謹慎檢查活動(Activities)狀態，是否如安全設計預期運作，防止意外外洩敏感性資料。

說明：

一般而言 Android APP 中的 Activity 是代表一個畫面。如果 Activity 是設定為 exported=true，那這個畫面就會被其他的程式存取，一般而言您往往不希望這個行為發生。

另外如果 Activity 有設定 Intent-filter 的時候，這個 Activity 就會自然變成公開且可被其他程式存取，就算是 Activity 並沒有設定為公開，他仍然可能會被其他程式啟動。這樣一來，可能會讓攻擊者可以在非程式開發者預期狀況外而執行程式。例如：有可能經由此種方式將密碼鎖檔的畫面直接跳過。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：Java

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.appsec.privateactivity" >

    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >

        <!-- Private activity -->

        <!–明文設定 exported attribute to false. -->
        <activity
            android:name=".PrivateActivity"
            android:label="@string/app_name"
            android:exported="false" />

        <!-- Public activity launched by launcher -->
        <activity
            android:name=".MyActivity"
            android:label="@string/app_name"
      android:exported="true" >
        <intent-filter>
           <action android:name="android.intent.action.MAIN" />
```

```
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
</manifest>

PrivateActivity.java
package org.appsec.privateactivity;

import android.view.View;
import android.widget.Toast;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;

public class PrivateActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.private_activity);

        // 謹慎處理接收到 Intent 的資訊，就算是從同一個應用程式傳來的
String data = getIntent().getStringExtra("Data");
    Toast.makeText(this, String.format("Received Data: \"%s\"", data),
Toast.LENGTH_LONG).show();
}
}

MyActivity.java
package org.appsec. privateactivity;

import android.view.View;
import android.widget.Toast;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
```

```
public class MyActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.my_activity);
    }

    public void onUserClick(View view) {

        //明訂同一應用程式內接收此 intent 的類別，
        Intent intent = new Intent(this, PrivateActivity.class);

// 此處因已確認接受類別為同一應用程式內，可傳遞敏感資料
        intent.putExtra("DATA", "Sensitive Data");
        startActivity (intent);
    }
}
```

行動應用 App 基本資安規範：4.1.5.1.防範惡意程式碼與避免資訊安全漏洞

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：JSSEC AASDSCG.4.1

參考來源：

how-to-export-an-activity-so-other-apps-can-call-it(查詢時間：2016/10/1)

CWE/OWASP

- M8 - Security Decisions via Untrusted Inputs(查詢時間：2016/10/1)
- M10 - Lack of Binary Protections(查詢時間：2016/10/1)
- CWE 927 (查詢時間：2016/10/1)

Android-03:Android 版行動應用 App 實作廣播(Broadcast intent)應設定權限，避免被惡意行動應用 App 偽造元件。

說明：

當發送廣播意圖(Intent)，如果沒有設定權限，那麼任何非特權應用程式皆可以接收意圖(Intent)，除非它有一個明訂的目標. 若有一個惡意的開發者創建具有完全相同的元件名的應用程式假裝為一個合法的元件，只要該命名空間尚未使用，此惡意程序將可以安裝在裝置上。

注意：

說定權限來保護應用程式的意圖(Intent)，並注意應通過廣播發送敏感資訊給第三方組件，因這個組件有可能會在惡意安裝中被替換而需注意。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：Java

Sender.java

```
// 發送廣播事件用的 Action 名稱
public static final String BROADCAST_ACTION =
        "org.appsec.broadcast.action.MYBROADCAST";
...
// 建立準備發送廣播事件的 Intent 物件
Intent intent = new Intent(BROADCAST_ACTION);
// 如果需要的話，也可以設定資料到 Intent 物件
intent.putExtra("UserName", userName);
intent.putExtra("UserId", userId);
// 發送廣播事件
sendBroadcast(intent);
```

Recevier.java

```
// 繼承自 BroadcastReceiver 的廣播接收元件
public class MyBroadcastReceiver extends BroadcastReceiver {
    // 接收廣播後執行這個方法

    @Override
    public void onReceive(Context context, Intent intent) {
        // 讀取包含在 Intent 物件中的資料
        String name = intent.getStringExtra("UserName");
        String id = intent. getStringExtra ("UserId");
        ...
        // 因為這不是 Activity 元件，需要使用 Context 物件的時候，
```

```
        // 不可以使用「this」，要使用參數提供的 Context 物件
        Toast.makeText(context, message, Toast.LENGTH_SHORT).show();
    }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application … >
        <!-- 使用 receiver 標籤，名稱設定廣播接收元件類別名稱 -->
        <receiver android:name="MyBroadcastReceiver">
            <intent-filter>
                <!-- 使用 Action 名稱設定接收的廣播事件 -->
                <action android:name=
                    " org.appsec.broadcast.action.MYBROADCAST" />
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

行動應用 App 基本資安規範：4.1.5.1. 防範惡意程式碼與避免資訊安全漏洞

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：JSSEC AASDSCG .4.2

參考來源：

Android：

- http://commonsware.com/blog/2013/09/11/beware-accidental-apis-avoid-intents-extras(查詢時間：2016/10/1)
- http://commonsware.com/blog/2014/04/30/if-your-activity-has-intent-filter-export-it(查詢時間：2016/10/1)
- https://www.appvigil.co/blog/2015/04/intent-spoofing-vulnerability-in-android-apps (查詢時間：2016/10/1)
- http://shop.oreilly.com/product/0636920022596.do(查詢時間：2016/10/1)

CWE/OWASP：

- M8 - Security Decisions via Untrusted Inputs(查詢時間：2016/10/1)
- M10 - Lack of Binary Protections(查詢時間：2016/10/1)
- CWE 925 (查詢時間：2016/10/1)

Android-04:Android 版行動應用 App 實作 PendingIntents，應指定為私有(Private)的廣播接收者/活動，明確在 base intent 指定元件名稱，避免接收外部惡意資料輸入。

說明：

PendingIntents 允許應用程式將執行作業傳遞到另一個應用程式，然後可以呼叫指定的意圖(Intent)，最終結果將類似原始應用程式的執行。這將允許外部應用程式呼叫到原應用程式內的私有元件。惡意外部應用程式可能試圖影響目標或其資料的完整性。

注意：

最理想的方式是把謹慎實作暫停意圖(PendingIntents)作為，遲發性 callback 私有廣播接收者或活動(private Broadcast Receivers or Activities)並在內部的意圖(Intent)明確指定呼叫元件的名稱。

安全開發生命週期：開發實作階段

不安全程式碼範例：Java

```
//implicit
Intent intent = new Intent("com.my.app.action") ;
PendingIntent pi = PendingIntent.getService(getApplicationContext(), 0, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
```

安全程式碼範例：Java

```
//explicit (to MyService)
Intent intent = new Intent(context, MyService.class);
PendingIntent pi = PendingIntent.getService(getApplicationContext(), 0, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
```

行動應用 App 基本資安規範：4.1.5.1. 防範惡意程式碼與避免資訊安全漏洞

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：JSSEC AASDSCG 4.7

參考來源：

Android：

- Sample code here https://gist.github.com/scottyab/d5ab6a284622ebc46d5a (查詢時間：2016/10/1)

CWE/OWASP：

- M8 - Security Decisions via Untrusted Inputs(查詢時間:2016/10/1)
- M10 - Lack of Binary Protections(查詢時間：2016/10/1)
- CWE 927(查詢時間：2016/10/1)

Android-05:Android 版行動應用 App 為避免內部的 Services 不被惡意外部行動應用 App 呼叫，可於 AndroidManifests.xml 設定存取權限保護。

說明：

服務(Service)通常用於背景處理。類似活動(Activity)和廣播接收器(Broadcast Receiver)，外部應用程式可直接呼叫服務，故應設定使用權限(Permission)和 export 屬性來提供適當的保護。

注意：

一個服務可以有一個以上的方法(Method)提供給外部程式呼叫。開發者可以定義服務的每個方法，並具有自己的權限(Permission)並使用 checkPermission 檢查呼叫端的程式具有相對應的權限。此外或者可以從系統設定方式著手，將一個服務的多個方法拆成多個服務，通過使用 AndroidManifest.xml 設定檔來定義每個服務的權限並由系統提供安全訪問。當呼叫需要傳送敏感資料的服務時，需要驗證呼叫的服務是正確的，而不是一個惡意服務。當開發者知道該元件的正確完整名稱時，應指定正確的名字給意圖(Intent)，再使用其作為連接到服務。或者服務可使用 checkPermission()再次以驗證有無特定的權限可收到預期的意圖。使用者在安裝時決定是否授予特定權限給安裝的應用程式。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：Java

//此範例為設定一自行定義的權限(Permission)，並使用它來限制存取 MyService

```
<permission android:name="com.example.mypermission" android:label="my_permission"
android:protectionLevel="dangerous">
</permission>`

<service
android:name="org.appsec.MyService" android:permission="org.appsec.mypermission">
<intent-filter>

<action android:name="org.app.MY_ACTION" />
</intent-filter>
</service>
```

行動應用 App 基本資安規範：4.1.5.1.防範惡意程式碼與避免資訊安全漏洞

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：JSSEC AASDSCG 4.4

參考來源：N/A

Android-06:Android 版行動應用 App 的敏感資料在不同行動應用 App 間傳遞時，避免使用廣播式 Intent，易遭惡意 App 讀取。

說明：

當一個應用程式使用一個廣播意圖(Broadcast Intent)來啟動一個 Activity 時，在意圖內傳遞的資料，有可能會被一個惡意的應用程式來讀取。惡意應用程式也可以讀取最近應用程式送出的一系列意圖。例如：當一個應用程式啟動 Android 網頁瀏覽器時，網址 URL 的資訊將會一其傳送過去，而這個網址的資料是可以側錄。

注意：

建議敏感資料不可以使用廣播意圖(Broadcast Intent)在應用程式之間傳遞。應該使用明訂式意圖(Explicit Intent)來取代，以提供安全性。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.5.1.防範惡意程式碼與避免資訊安全漏洞

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：JSSEC AASDSCG 4.7

參考來源：CWE/OWASP：

- M8 - Security Decisions via Untrusted Inputs(查詢時間：2016/10/1)
- M10 - Lack of Binary Protections(查詢時間：2016/10/1)
- CWE 285(查詢時間：2016/10/1)

Android-07: Android 版行動應用 App 應謹慎實作 Content Providers，避免因權限過高或未驗證資料目的地與來源，遭惡意程式注入式攻擊。

說明：

內容提供者(Content Providers)是一個標準的方式讓應用程式可以使用 URI 地址定義來分享資料和關連式資料庫的模型。也可以用來當做使用 URI 地址定義來存取檔案的方式。

注意：

內容提供者(Content Providers)可以宣告權限(Permissions)，並且可以限制讀和寫的權限。除非有其必要性，請勿允許寫入權限。同理除非必要，請設定讀取限制，以免未授權的應用程式可以讀取內容提供者，以提高安全性。

內容提供者也允許資料紀錄等級(Record Level)的分享而不用整個資料庫分享，使用此特性來提供操作上需要的最低的存取限制。例如：可以透過傳送一筆即時訊息給外部程式，讓外部程式將訊息以 Email 發送。當外部程式結束後，這個派送就自動結束。

傳遞給內容提供者的參數需視為不受信任的輸入，不可以在未經正規化過濾，就直接組合成 SQL 查詢。SQL 程式碼可以在內容提供者的請求中被送出，並且如果此 SQL 程式碼是一個查詢，則內容提供者會返回結果資料或控制給攻擊者。內容提供者的服務是基於檔案層級時，要確保文件名稱的路徑遍歷有過濾掉。例如：如果攻擊者使用“../../../file”作為檔案名時，它可能導致程序讀取非預設的文件目錄，並且此資料將會被攻擊者得到。另外注意，如果攻擊者建立符號鏈接(Symbolic Link)也可能產生類似的結果。

安全開發生命週期：開發實作階段

不安全程式碼範例： Java

SQL 語法直接使用+的法式將使用者的輸入串連起來

```
@Override
public Cursor query(Uri uri, String[] projection, String selection, String[]
selectionArgs, String sortOrder) {
        String groupId = getFromUserInput();//假設有此方法可讀入畫面上使用者的輸入值
        SQLiteDatabase mDB = mHelper.getWritableDatabase();
        Cursor out = mDB.rawQuery("SELECT * FROM product where group_id="+groupId,
null);
        out.moveToFirst();
        return out;
}
```

安全程式碼範例： Java

使用參數化方式，將 SQL 語法和使用者的輸入分離

```
@Override
public Cursor query(Uri uri, String[] projection, String selection, String[]
selectionArgs, String sortOrder) {
        String groupId = getFromUserInput();//假設有此方法可讀入畫面上使用者的輸入值
        SQLiteDatabase mDB = mHelper.getWritableDatabase();
        Cursor out = mDB.rawQuery("SELECT * FROM product where group_id=?", new
String[]{groupId});
        out.moveToFirst();
        return out;
}
```

行動應用 App 基本資安規範：4.1.5.1.防範惡意程式碼與避免資訊安全漏洞

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：JSSEC AASDSCG .4.3

參考來源：CWE/OWASP：

- M7 - Client Side Injection(查詢時間：2016/10/1)
- CWE 926(查詢時間：2016/10/1)

Android-08:Android 版行動應用 App 應謹慎實作檔案權限，除非有必要為任何應用程式建立可以任意讀取或寫入檔案，避免創造全域可讀寫檔案權限，導致敏感性資料外洩。

說明：

無設權限讀取檔案可能會造成敏感資料的外洩。

無設權限寫入檔案可能會造成第三方程式可藉由此讀取任意資料庫，以影響其行為，例如：系統設定檔或登入資訊。

注意：

不可將檔案的權限設定為 MODE_WORLD_READABLE or MODE_WORLD_WRITABLE,就算是檔案存在程式私有的目錄，除非檔案真的需要被任何 APP 存取。

不可使用 chmod or syscalls 來改變檔案的屬性為 0666, 0777, 0664。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：4.1.5.1.防範惡意程式碼與避免資訊安全漏洞

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：JSSEC AASDSCG.4.5,4.6

參考來源：N/A

Android-09:行動應用 App 應實作過濾使用者輸入及伺服器端傳入資料中易導致 Intent Injection 之字串。

說明：

防止意圖欺騙：

1.限制通訊

Android 平台有兩個控制元件限制可以控制權限使用相對應權限的應用程式，以防止與應用程式通信惡意應用，以及意圖類型使用隱式意圖產生更多的安全問題。

2.驗證輸入並假定輸入是不一定是來自可信賴的來源以及驗證每個應用程式的輸入。

3.設定值在外部的 Android 的 AndroidManifest 設定值：避免接收到惡意的意圖出口組件。

安全開發生命週期：開發實作階段

不安全程式碼範例：Java

```
TextView tv = (TextView) findViewById(R.id.textview);
InputStreamReader isr = null;
char[] text = new char[1024];
int read;
try {
    String urlstr = getIntent().getStringExtra("WEBPAGE_URL");
    URL url = new URL(urlstr);
    isr = new InputStreamReader(url.openConnection().getInputStream());
//此處並無防止"file://..." 這種型式的。在此情形之下，本機端的檔案會被開啟，並且將內容秀在
TextView 中，而非原先預請的遠端網頁

    while ((read=isr.read(text)) != -1) {
        tv.append(new String(text, 0, read));
    }
} catch (MalformedURLException e) { ...
```

安全程式碼範例：Java

```
TextView tv = (TextView) findViewById(R.id.textview);
InputStreamReader isr = null;
char[] text = new char[1024];
int read;
try {
    String urlstr = getIntent().getStringExtra("WEBPAGE_URL");
    URL url = new URL(urlstr);
    //加入明確檢查是否是使用 http 或 https protocol, 如不是則產生例外
    String prot = url.getProtocol();
    if (!"http".equals(prot) && !"https".equals(prot)) {
```

```
        throw new MalformedURLException("invalid protocol");
    }
    isr = new InputStreamReader(url.openConnection().getInputStream());
    while ((read=isr.read(text)) != -1) {
        tv.append(new String(text, 0, read));
    }
} catch (MalformedURLException e) { ..
```

行動應用 App 基本資安規範：4.1.5.4.使用者輸入驗證

行動應用 App 基本資安檢測基準：

4.1.5.4.2.(7)檢查行動應用程式是否過濾導致 Intent Injection 之字串。

各國開發要點參考：JSSEC AASDSCG 3.2

參考來源：N/A

Android-10:Android 版本行動應用 App 的 Webview 的許多 API 已被發現漏洞，如 addJavscriptInterface ，實依最佳安全實務實作 WebView。

說明：

網頁視圖元件(WebView)在使用上可能導致一些安全問題，在實作上應注意避免資安問題。特別是有許多利用 addJavscriptInterface() API 而產生已知的資安漏洞。

注意：

如果不需要使用 JavaScript 和插件，請將網頁視圖元件(WebView)設定為禁用 JavaScript 和插件的支持。預設值是禁用，但好的習慣是，明確設定為禁用。

設定禁用本地文件訪問。如此可限制訪問應用程式的資源和資產目錄(Resource and Asset Directory)，避免從網頁的攻擊，試圖獲得其他本地訪問文件資料。

防止從第三方主機上傳內容。要從一個應用程式內完全阻止這是有難度的，但開發人員可以重寫 shouldOverrideUrlLoading 和 shouldInterceptRequest 來攔截，檢查和驗證從網頁視圖元件(WebView)內發起的大多數請求。白名單機制，也可以使用 URI 類別來檢查 URI 的組成部分，並確認有列在核定資源的白名單內。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：Java

SaferWebViewClient.java

```
/**
   * Implements whitelisting on host name
   */
public class SaferWebViewClient extends WebViewClient {

       private String[] hostsWhitelist;

       public SaferWebViewClient(String hostsWhitelsit){
           super();
           this.hostsWhitelist = hostsWhitelist;
       }

       @Override
       public WebResourceResponse shouldInterceptRequest(final WebView view, String
url) {
           if (isValidHost(url)) {
               return super.shouldInterceptRequest(view, url);
           } else {
               return getWebResourceResponseFromString();
           }
```

```
        }
private WebResourceResponse getWebResourceResponseFromString() {
            return getUtf8EncodedWebResourceResponse(new
StringBufferInputStream("alert('!NO!')"));
        }

        private WebResourceResponse getUtf8EncodedWebResourceResponse(InputStream
data) {
            return new WebResourceResponse("text/css", "UTF-8", data);
        }



        @Override
        public boolean shouldOverrideUrlLoading(WebView view, String url) {
            return isValidHost(url);
        }

        private boolean isValidHost(String url){
            if (!TextUtils.isEmpty(url)) {
                final String host = Uri.parse(url).getHost();
                for (String whitelistedHost: hostsWhitelist){
                    if (whitelistedHost.equalsIgnoreCase(host)){
                        return true;
                    }
                }
            }
            return false;
        }
}
WebviewUtil.java
public class Util {
public static void disableRiskySettings(WebView webView){

        //javascript could be a vector to exploit your applications
        webView.getSettings().setJavaScriptEnabled(false);

        //default is off, but just in case. plugins could be a vector to exploit your
applications process
```

```
        webView.getSettings().setPluginState(WebSettings.PluginState.OFF);

        //Should an attacker somehow find themselves in a position to inject script
into a WebView, then they could exploit the opportunity to access local resources.
This can be somewhat prevented by disabling local file system access. It is enabled
by default. The Android WebSettings class can be used to disable local file system
access via the public method setAllowFileAccess.
        //This restricts the WebView to loading local resources from
file:///android_asset (assets) and file:///android_res (resources).
        webView.getSettings().setAllowFileAccess(false);

        //disable Geolocation API
        webView.getSettings().setGeolocationEnabled(false);

    }

}
```

行動應用 App 基本資安規範：4.1.5.1.防範惡意程式碼與避免資訊安全漏洞

行動應用 App 基本資安檢測基準：

4.1.5.4.2.(2)檢查行動應用程式是否過濾導致 JavaScript Injection 之字串。

各國開發要點參考：JSSEC AASDSCG 4.9

參考來源：N/A

3.3.iOS 安全開發實務

iOS-01:謹慎使用 Keychain 儲存密碼(Use the Keychain carefully)。

說明：

- 為保護資料儲存的安全性，iOS 作業系統提供了鑰匙串(Keychain)功能。然而在一些情況下，Keychain 可能會受到損害並遭到解密。
  - 使用 iOS 7 以上的版本時，若惡意第三方有機會存取到有加密的 iTunes 備份，Keychain 就有可能被破解。因為當 iTunes 備份被啟用時，iOS 會重新對 Keychain 進行加密，所以這時如果惡意第三方可以知道備份的加密密碼，那麼 Keychain 就有可能部分被解密。
  - 此外，Keychain 在有進行越獄(jailbreak)的設備上，其存取控管也可能會失效。在有越獄的手機上，任何的應用程式都可能存取到別隻程式的 Keychain 內容。而對於那些本身就含有 Bootrom 漏洞(Bootrom exploit)的舊設備(例如：iPhone4)，攻擊者可以藉由實體存取而破解 Keychain。
- 當有使用 Keychain 儲存資料時，開發人員應使用最嚴格的防護類別(可參考 kSecAttrAccessible 屬性)，而且使用該種類別(class)完全不會影響到 App 本身的運作流暢度。例如：若你的應用程式並不是設計於背景(background)執行的，可使用 kSecAttrAccessibleWhenUnlocked 或 kSecAttrAccessibleWhenUnlockedThisDeviceOnly。
- 若要避免因 iTunes 備份讓 Keychain 曝光，可以使用「ThisDeviceOnly」保護類別。
- 對於高度敏感的資料，可考慮使用 Keychain 所提供的另一種更安全的保護機制，即應用程式層的加密機制。例如：在進入應用程式時，使用者會輸入通行密碼(passphrase)以進行認證，並在資料儲存到 Keychain 前，用此通行密碼對資料進行加密。

安全開發生命週期：開發實作階段

不安全程式碼範例：

使用其他屬性的參數，造成可能的漏洞，例如：備份檔被其他機器還原。

安全程式碼範例： Objective-C

使用嚴格參數，以免資料洩漏

```
// Protect the keychain entry so it's only valid when the device is unlocked.
 [dictionary setObject:(__bridge
id)kSecAttrAccessibleWhenUnlockedThisDeviceOnly forKey:(__bridge
id)kSecAttrAccessible];
```

行動應用 App 基本資安規範：4.1.2.3. 敏感性資料儲存

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源

iOS：Keychain Services Programming Guide (查詢時間：2016/10/1)

CWE/OWASP：

- M2 - Insecure Data Storage(查詢時間：2016/10/1)
- M5 - Poor Authorization and Authentication(查詢時間：2016/10/1)
- CWE 312, 522(查詢時間：2016/10/1)

iOS-02:為保護敏感資料不被以截圖方式儲存於檔案系統，行動應用 App 使用 API 設定或編寫程式阻擋敏感資料區快照功能(Snapshots)或以覆蓋方式清除。

說明：

為了提供在界面上的 APP 切換，iOS 裝置已被證明在切到背景時會儲存螢幕快照。而這個快照會被存放在一個資料夾，將可能衍生安全問題。為了保護敏感資料，於螢幕切換時進行畫面保護，或隱藏安全疑慮的欄位是建議的安全方法。

安全開發生命週期：開發實作階段

不安全程式碼範例：N/A

安全程式碼範例：Objective-C

- *提供蓋掉螢幕的範例碼、提供隱藏欄位的建議方法*
- *先製作滿版圖片 secure-image.png (320x568 for iPhone)*

```
- (void)applicationDidEnterBackground:(UIApplication *)application {
    // Use this method to release shared resources, save user data, invalidate timers, and store enough application state information to restore your application to its current state in case it is terminated later.
    // If your application supports background execution, this method is called instead of applicationWillTerminate: when the user quits.
        if (!self.backgroundImage){
            UIImageView *myBanner = [[UIImageView alloc] initWithImage:[UIImage imageNamed:@"secure-image.png"]];
            self.backgroundImage = myBanner;
        }
        [self.window addSubview:self.backgroundImage];
    // you can hide security field here
    // 隱藏欄位
    // [viewControl.secure_field setHidden:TRUE];
}
- (void)applicationWillEnterForeground:(UIApplication *)application {
    // Called as part of the transition from the background to the inactive state; here you can undo many of the changes made on entering the background.
    if (self.backgroundImage)
        [self.backgroundImage removeFromSuperview];
    // you should visi security field here
    // 回復已隱藏欄位
    // [viewControl.secure_field setHidden:FALSE];
}
```

行動應用 App 基本資安規範：4.1.2.3. 敏感性資料儲存

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：

Managing Your Applications Flow（查詢時間：2016/10/1）

CWE/OWASP：

- M4 - Unintended Data Leakage（查詢時間：2016/10/1）
- M2 - Insecure Data Storage（查詢時間：2016/10/1）
- CWE 200（查詢時間：2016/10/1）

iOS-03:啟用自動引用計數(Automatic Reference Counting ,ARC)，避免記憶體物件弱點產生。

說明：

自動引用計數(ARC)是一個記憶體管理系統，在編譯/執行時會自動處理，而不是交由開發者的引用計數。此功能在 iOS5 時推出，但也允許使用者自行管理記憶體釋放。而進一步，推出 Switf 語言時，物件與物件的強型鏈結，也造成需要注意的 ARC 項目。

【不啟用 ARC】：當不啟用 ARC 的專案，記憶體需自行釋放。

【啟用 ARC】：專案啟用 ARC 時，可定義編譯參數，以允許某些檔案-不啟用 ARC。

強制關閉某 Class 方法：在專案→Build Phases→Compile Sources 設定 Compiler Flags 增加[-fno-obj-arc]

安全開發生命週期：開發實作階段

不安全程式碼範例：Objective-C

不啟用 ARC 時，又沒有完成釋放記憶體。

```
@interface myInputStream : NSObject {
    NSInputStream *stream;
}
+ (void)initialize
{
    stream = NSInputStream inputStreamWithFileAtPath:somepath]];
}
- (void)dealloc
{
        //需要手動release
    //[stream release];
    [super dealloc];
}
```

Swift

Swift 循環強型鏈結問題

物件宣告

```
class Person {
    let name: String
    init(name: String) { self.name = name }
    var apartment: Apartment?
     deinit { print("\(name) is being deinitialized") }
}
class Apartment {
    let unit: String
    init(unit: String) { self.unit = unit }
    var tenant: Person?
    deinit { print("Apartment \(unit) is being deinitialized") }
}
```

參數宣告

```
var john: Person?
var unit4A: Apartment?
```

建立實體

```
john = Person(name: "John Appleseed")
unit4A = Apartment(unit: "4A")
```

關係指派

```
john!.apartment = unit4A
unit4A!.tenant = john
```

釋放實體

```
john = nil
unit4A = nil
```

強型鏈結而導致 ARC 無法正確運作，記憶體無法被釋放。

安全程式碼範例： Swift

循環強型鏈結問題，改用弱型鏈結

```
class Apartment {
    let unit: String
    init(unit: String) { self.unit = unit }
    weak var tenant: Person?
    deinit { print("Apartment \(unit) is being deinitialized") }
}
```

行動應用 App 基本資安規範：4.1.5.1.防範惡意程式碼與避免資訊安全漏洞

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：Swift Programming Language - AutomaticReferenceCounting - iOS developer(查詢時間：2016/10/1)

iOS-04:啟用 App Transport Security(ATS)設定。

說明：

因應 App 網路需求，從 iOS 9.0 開始，增加應用程式傳輸安全(ATS)的新安全功能，並預設啟用。重新編譯的 App 就需考慮此設定，否則會造成無法使用非 HTTPS 的網路資源。

Apple 宣布 2017 年開始，所有 App 的資料連結必須啟用 HTTPS 才能上架。

藉此要求提高了對執行基於 HTTP 的請求增加額外的安全性要求，並保護應用程式和網路服務之間的連接的保密性和資料完整性。ATS 強調 HTTP 連接必須使用 HTTPS(RFC 2818)。HTTPS 請求必須使用安全通信的最佳實作，即 TLS 1.2(RFC 5246)。

ATS 是由 NSURLSession 類和使用它的所有 API 執行。舊 NSURLConnection 的元件也受 ATS 管制。

參數說明：主目錄名稱 NSAppTransportSecurity，類型：Dictionary。

第一層：第一層有兩種可設定種類

名稱	類型	描述
NSExceptionDomains	Dictionary	可選的參數 針對每個網域，定義該網域的設定值 例如：google.com 或者 edition.cnn.com
NSAllowsArbitraryLoads	Boolean	允許任何網站的載入。 其他網域的 ATS 設定。 預設值是 NO，如果改為 YES，則會關閉其他網域的 ATS 機制(即允許 HTTP)。

第二層：每個 NSExceptionDomains，需自定義，格式如

名稱	類型	描述
NSIncludesSubdomains	Boolean	可選布林值，預設 NO。 若設定 YES：代表此設定應用至其子網域。例如：YES 代表影響 *.google.com
NSExceptionAllowsInsecureHTTPLoads	Boolean	可選布林值，預設 YES。 若設定 YES：代表允許此網域可使用非安全載入。
NSExceptionRequiresForwardSecrecy	Boolean	可選布林值，預設 YES。 若設定 YES：代表 HTTPS 的公鑰傳輸保密必需符合 forward secrecy (FS)的某一項；若設定為 NO，則代表放寬規則(也允許以下規格)

名稱	類型	描述
		TLS_RSA_WITH_AES_256_GCM_SHA384 TLS_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_AES_256_CBC_SHA256 TLS_RSA_WITH_AES_256_CBC_SHA TLS_RSA_WITH_AES_128_CBC_SHA256 TLS_RSA_WITH_AES_128_CBC_SHA
NSExceptionMinimumTLSVersion	String	可選字串值，預設 TLSv1.2。此值定義可接受最低版本的 Transport Layer Security (TLS) 來進行連線。 可接受的值如下： TLSv1.0 TLSv1.1 TLSv1.2
NSThirdPartyExceptionAllowsInsecureHTTPLoads	Boolean	類似於 NSExceptionAllowsInsecureHTTPLoads 項目，對象是 App 應用的第三方元件的網域所需要的 TLS 的最低版本。
NSThirdPartyExceptionRequiresForwardSecrecy	Boolean	類似於 NSExceptionRequiresForwardSecrecy 項目，對象是 App 應用的第三方元件的網域所需要的 TLS 的最低版本。
NSThirdPartyExceptionMinimumTLSVersion	String	類似於 NSExceptionMinimumTLSVersion 項目，對象是 App 應用的第三方元件的網域所需要的 TLS 的最低版本。

當 ATS 機制運作下，網路連結的要求必須符合以下每一項目

1.網站主機的憑證必須符合以下其中一項的信任需求

a.由某個憑證當局 certificate authority(CA)所發給，該根憑證有內含於作業系統。

b.由信任的 root CA，並且由系統管理員做匯入。

2.交談的安全層協議必須是 TLS 1.2

3.交談的 TLS 密碼套件必須支援 forward secrecy (FS)，並且是其中以下一項：

- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

4.子伺服器憑證必須由以下的金鑰簽章。

- Rivest-Shamir-Adleman (RSA) key with a length of at least 2048 bits
- Elliptic-Curve Cryptography (ECC) key with a size of at least 256 bits

並且，子伺服器憑證雜湊演算法必須符合 SHA-2/SHA-256 以上。

當 ATS 機制不運作之下，網路連結仍然被程式撰寫的程式碼控制。當 ATS 機制運作下，你的程式無法覆蓋 ATS 所設定之規則。

安全開發生命週期：開發實作階段

不安全程式碼範例：

範例一：全面取消 ATS

```
NSAppTransportSecurity

    NSAllowsArbitraryLoads = YES
```

範例二：某網域降級 TLS

```
NSAppTransportSecurity

    NSExceptionDomains

        "less-secure.example.com"

            NSExceptionRequiresForwardSecrecy = NO

            NSExceptionMinimumTLSVersion = "TLSv1.0"
```

範例三：某網域取消 ATS

```
NSAppTransportSecurity

    NSAllowsArbitraryLoads = NO  // Shown for clarity; this is the default

    NSExceptionDomains

        "secure-server-i-control.example.com"

        NSExceptionAllowsInsecureHTTPLoads = YES

        NSExceptionRequiresForwardSecrecy = NO

        NSExceptionMinimumTLSVersion = "TLSv1.0"
```

安全程式碼範例：N/A

行動應用 App 基本資安規範： 4.1.2.4.敏感性資料傳輸

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：N/A

3.4.伺服器安全實務

Server-01 與行動應用 App 連接之所有後端服務伺服器(包含網頁、資料庫及中介等)作業系統應有效強化及進行安全設定配置，並持續進行安全性程式修補。

說明：

針對 Server 部分，因應各種資安漏洞依每項問題討論如下

【減少資訊洩露】

在 Web 伺服器上的某些設置可以提高安全性，當 Web 伺服器上被忽視的脆弱性資訊被披露，此資訊披露可能會導致嚴重的問題，因為攻擊者可以獲得片段有用資訊而更容易攻擊伺服器。

減少資訊洩露是簡單又安全的原則，錯誤訊息揭露在開發環境中是有用的，但在正式環境中就是資料洩漏，例如：網頁的元件、版本、作業系統等。攻擊者可以使用該訊息配合相關的漏洞、缺陷以達成攻擊的目的。

建議例如：降低 Apache 的版本資料，或務必刪除某些預設的路徑，或特別的安裝路徑。

另一方面，管理者功能路徑，除非必要，否則不應提供公開存取(加入安全限制)。

【強制 HTTPS 機制】

HTTP Strict Transport Security (HSTS) (查詢時間：2016/10/1)

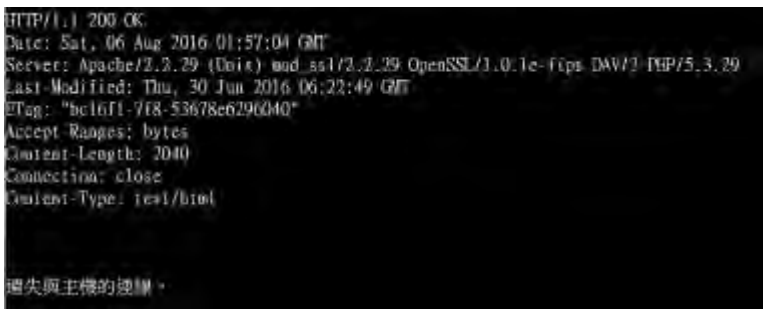
於網頁主機加上此機制(在 header 加上"Strict-Transport-Security")，則使用者端的瀏覽器，就能配合強制鎖定 HTTPS，以保護此連線。

安全開發生命週期：部署維運階段

不安全程式碼範例：網站主機：Server

例如：過時 OpenSSL 版本 OpenSSL/1.0.1e 與太多資訊洩漏。

```
telnet ---.---.---.tw 80
Trying 0.0.0.0……....
Connected to ---.---.---.tw.
Escape character is '^]'.
HEAD / HTTP/1.0
```



安全程式碼範例：

網站主機：Server

較少資訊洩漏

```
telnet www.-----.com.tw 80

HTTP/1.0 200 OK

Date: Sat, 30 Jul 2016 05:52:56 GMT

P3P: policyref="http://info.-----.com/w3c/p3p.xml", ... Cache-Control: max-age=3600, public

Vary: Accept-Encoding

Content-Type: text/html; charset=UTF-8

Server: ATS
```

網站主機：Server .NET

啟用 HSTS 機制

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <system.webServer>
        <rewrite>
            <rules>
                <rule name="HTTP to HTTPS redirect" stopProcessing="true">
                    <match url="(.*)" />
                    <conditions>
                        <add input="{HTTPS}" pattern="off" ignoreCase="true" />
                    </conditions>
                    <action type="Redirect" url="https://{HTTP_HOST}/{R:1}"
                        redirectType="Permanent" />
                </rule>
            </rules>
            <outboundRules>
                <rule name="Add Strict-Transport-Security when HTTPS" enabled="true">
                    <match serverVariable="RESPONSE_Strict_Transport_Security"
                        pattern=".*" />
                    <conditions>
                        <add input="{HTTPS}" pattern="on" ignoreCase="true" />
                    </conditions>
                    <action type="Rewrite" value="max-age=31536000" />
                </rule>
            </outboundRules>
```

```
      </rewrite>
   </system.webServer>
</configuration>
```

行動應用 App 基本資安規範：N/A

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 5.3

參考來源：

網站主機：

- making-http-requests-via-telnet/(查詢時間：2016/10/1)
- enable-http-strict-transport-security-hsts-in-iis-7(查詢時間：2016/10/1)

Server-02:應依 CPD-01 需求保留日誌在安全且不易遭非授權存取或場篡改儲存空間，作為後續突發事件反映及數位取證原始資料。

說明：

確保有足夠的日誌保留在後台以檢測和對突發事件作出反應，並進行取證(個人資料保護法的範圍內)。

安全開發生命週期：部署維運階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：N/A

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG5.4

參考來源：N/A

Server-03:實施 code review，檢查在行動智慧裝置和網頁伺服器端和其他外部介面之間傳送的有無任何意外敏感性資料傳輸。

說明：

貫徹對無意傳送敏感資料的程式碼的特定檢查，在行動智慧裝置和網路後端伺服器和其他外部介面之間傳送的有無任何意外敏感性資料傳輸(例如：是位置或包含的文件的 metadata 中的其他資訊)。

安全開發生命週期：測試階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範： N/A

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 5.1

參考來源：N/A

Server-04:與行動應用 App 連接之所有後端服務(Web 服務/REST)應定期檢測漏洞，如使用靜態程式碼分析器工具測試和模糊工具檢測和發現的安全漏洞，並及時修復高風險漏洞。

說明：

行動應用程式所有後端服務(Web 服務/REST)應定期檢測漏洞，例如：使用靜態程式碼分析器工具測試和模糊工具檢測和發現的安全漏洞。

安全開發生命週期：部署維運階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：N/A

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 5.2

參考來源：N/A

Server-05:與行動應用 App 連接之所有後端服務(Web 服務/REST)應定期實施滲透測試，發現安全漏洞，並及時修復高風險漏洞。

說明：

保護 Web Service 可藉由第一次上線或定期進行滲透測試(Protect and Pen Test Web services)以發現程式碼檢查無法發現之漏洞，並在合理的時間依風險高低順序修補。

安全開發生命週期：部署維運階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範： N/A

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：N/A

Server-06:網頁伺服器需預防網頁掛馬及點擊劫持攻擊。

說明：

網頁掛馬(Framing)使用 iFrame 來遞送一個 Web/WAP 網站連線。這種攻擊可以用"包裝(wrapper)"網站來執行點擊劫持(clickjacking)攻擊。點擊劫持是利用熱門的網站服務(例如：Facebook、Google、Twitter 等)，通過諸如跨網站程式碼(XSS Scripting)漏洞鏈受掌握受影響的主機控制權，於頁面上嵌入 iFrame 或程式碼。此一網頁頁框的主要目的是誘騙使用者點擊嵌在 iFrame 的隱藏連結，將使用者重導向到攻擊者控制的網站以竊取資訊的一個種威脅。而攻擊者的目標是蒐集機密資訊。這種攻擊通常需要嵌入程式碼，這是不在一般使用者的知識範圍內執行腳本的形式，可以作為攻擊使得使用者單擊顯示執行其他功能的按鈕來觸發。

為防止 Framing 做法的 iOS 最好的辦法是不使用網頁視圖(Web View)。為預防 Framing 機制之一是利用 client 端 JavaScript。大多數 Web 網站不再設計或禁止 JavaScript 運行，所以在 JavaScript 的安全措施(停用)是一個選項。雖然客戶端將可能受到影響，這層確實引起了攻擊者注目。下面是一個例子，迫使已上傳 JavaScript code 的 Frame 被破壞而無法執行。

透過額外的步驟，攻擊者可以添加到他們的頁框以試圖防止框架無效化程式碼，例如：警示使用者要求他們不退出。更複雜的 JavaScript 可能可以應對這種技術。至少基本頁框破壞程式碼(frame busting code)的加入使得簡單 Framing 更為艱難的過程。

X FRAME-OPTIONS 標頭為一個新的、更好的抗 Framing 選項最近在一些瀏覽器中達成的基礎上，回應發送的 HTTP 標頭。通過配置在 Web 伺服器等級這個標頭，瀏覽器被指示不顯示在一個框架或 iframe 的回應內容。在 Apache 的配置文件這樣的例子實作中提供的程式碼範例。

專為設計的 WebView 的 API 可以被濫用來破解的 WebView 中指定的 Web 內容的安全性。要防止這種眾所周知的漏洞的應用程式及其使用者的最佳方式是：

- 阻止上傳頁框託管於其他域名的請求內容的 X-Frame-Option HTTP response header。然而，與已被入侵主機連線時，這種緩解方法並不適用。
- 利用內部防禦機制，以確保所有 UI 元素在頂層框架上傳；這樣就避免了在較低的層級水準，通過不信任的頁框設置服務內容。

安全開發生命週期：部署維運階段

不安全程式碼範例：N/A

安全程式碼範例：

網頁主機：Server

增加 X-Frame-Options

```
X-Frame-Options: DENY

X-Frame-Options: SAMEORIGIN

X-Frame-Options: ALLOW-FROM https://example.com/
```

DENY

The page cannot be displayed in a frame, regardless of the site attempting to do so.

SAMEORIGIN

The page can only be displayed in a frame on the same origin as the page itself.

ALLOW-FROM *uri*

The page can only be displayed in a frame on the specified origin.

行動應用 App 基本資安規範： N/A

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：網頁主機：X-Frame-Options - mozilla developer(查詢時間：2016/10/1)

Server-07:網頁伺服器使用 Token 防範跨站請求(Cross-site Request Forgery, CSRF)攻擊。

說明：

跨站請求偽造(CSRF)是一款迫使終端使用者在他們目前正在驗證的 Web 應用程式執行不必要的行動的攻擊。CSRF 攻擊專門針對網站資料修改的目的，而非盜竊使用者資料，因為攻擊者並沒有看到這些資料。隨著社群功能的幫助之下(如發送電子郵件或聊天的連結)，攻擊者可能誘騙網站使用者執行攻擊者選擇的行為。如果受害者是普通使用者，一個成功的 CSRF 攻擊可以強制使用者執行像轉移資金、改變他們的電子郵件地址等的請求。如果受害者是管理權限帳戶，CSRF 攻擊將危及整個網站安全。

安全開發生命週期：開發實作、部署維運階段

不安全程式碼範例：N/A

安全程式碼範例：網站主機：.NET C#

使用 CSRF Token

1.網頁方式：在表單內，增加 TOKEN。

```
@using (Html.BeginForm("Manage", "Account")) {
    @Html.AntiForgeryToken()
}
```

2.Ajax 方式：在呼叫端，增加 TOKEN。

```
<script>
    @functions{
        public string TokenHeaderValue()
        {
            string cookieToken, formToken;
            AntiForgery.GetTokens(null, out cookieToken, out formToken);
            return cookieToken + ":" + formToken;
        }
    }

    $.ajax("api/values", {
        type: "post",
        contentType: "application/json",
        data: {  }, // JSON data goes here
        dataType: "json",
        headers: {
            'RequestVerificationToken': '@TokenHeaderValue()'
        }
    });
</script>
```

3.在主機接收端：驗證此次 TOKEN。

```
void ValidateRequestHeader(HttpRequestMessage request)
{
    string cookieToken = "";
    string formToken = "";

    IEnumerable<string> tokenHeaders;
    if (request.Headers.TryGetValues("RequestVerificationToken", out tokenHeaders))
    {
        string[] tokens = tokenHeaders.First().Split(':');
        if (tokens.Length == 2)
        {
            cookieToken = tokens[0].Trim();
            formToken = tokens[1].Trim();
        }
    }
    AntiForgery.Validate(cookieToken, formToken);
}
```

行動應用 App 基本資安規範：N/A

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：

網站主機：

- XSRF/CSRF Prevention in ASP.NET MVC and Web Pages - .NET microsoft（查詢時間：2016/10/1）
- Preventing Cross-Site Request Forgery (CSRF) Attacks in ASP.NET Web API（查詢時間：2016/10/1）

Server-08:實作網頁伺服器服務安全一般控管原則。

說明：

1. 當登入失敗，為避免資料洩漏。回應訊息不可洩漏帳號存在與否，應提示如：『帳號或密碼不正確，請重新登入！』。
2. 應加入防止機器人的機制，例如：google reCAPTCHA。
3. 密碼應要求至少 8 至 12 碼以上長度，甚至要求大小寫、特殊字元、數字等複雜組合。
4. 以帳號或 IP 為偵測原則，拒絕短時間錯誤登入，以防止惡意登入。
5. 更進一步：拒絕簡單密碼、要求每段時間更改密碼、新密碼不能與近期重複、拒絕常用組合。
6. 資料庫不直接儲存密碼，改以 HASH 儲存，並記得加上複雜的 Salt。

安全開發生命週期：部署維運階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：N/A

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：N/A

Server-09:應評估服務容量及水準，限制個別使用者/IP 流量，以降低被分散式阻斷攻擊(DDoS)風險。

說明：

開發人員應在設計階段估算未來正式運作時的伺服器承載能力，並限制每個個別使用者的合理流量，共在正式上線後，定期監控伺服器效能，以降低被被分散式阻斷攻擊(DDoS)風險。

安全開發生命週期：部署維運階段

不安全程式碼範例：N/A

安全程式碼範例：N/A

行動應用 App 基本資安規範：N/A

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：ENISA SSDG 5.5

參考來源：N/A

Server-10:參考 CPG-01~CPG-03 實作 TLS 伺服器端設定。

說明：

SSL 與 TLS 介紹

- SSL(Secure Sockets Layer 的簡寫)安全通訊協定，起源於網景公司(Netscape)在 1994 年推出首版網頁瀏覽器，網景領航員時，推出 HTTPS 協定，以 SSL 進行加密。歷經 SSL 1.0/SSL 2.0/SSL 3.0。於 2006 年 IETF 將 SSL 標準化，即 RFC 2246 ，並將其稱為 TLS 傳輸層安全協議(Transport Layer Security)，並歷經改版，目前符合公認安全的版本為 TLS 1.2 以上。

定義	SSL 1.0	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3
年份	NA	1995	1996	1999	2006	2008	待定

- 『SSL/TLS 更新歷程』。
- 『停用 1024 位元金鑰』。
- 美國國家標準技術研究所 NIST 建議於 2013 之前，金鑰長度由 1024 位元，轉換成 2048 位元。因 1024 位元金鑰有可能會被破解，而 2048 位元金鑰於理論上並不會在有限的時間(數年內)被破解，所以可信任。
- 『停用 SSL 3.0』。
- 自從 2014 SSL 3.0 的 CVE- 2014-3566 漏洞，會造成資料不安全，各大網站推廣停用 SSL3.0，改用 TLS 加密協定。

『OpenSSL heartbleed bug』

- OpenSSL 大量被應用於網頁主機上，於 2014/4/08 發布 1.0.1g 修正此問題。此問題會影響包括 SSL 與 TLS，會導致主機憑證私鑰被竊、主機資料被竊、客戶端資料被竊。
- 修正：主機端，需更新有問題的 OpenSSL 程式版本、更換新版的憑證。
- 修正：客戶端，需更新有問題的 OpenSSL 程式版本。
- 如何檢查版本：執行命令 openssl version
  得到輸出：OpenSSL 1.0.2 22 Jan 2015
- 驗證：2048 位元、SHA2 簽章、加密。

網站 SSL 憑證介紹與驗證

名稱	英文	年限	審查	差別
單機版	Standard SSL (DV)	1~3 年	網站管理者	標準保護

名稱	英文	年限	審查	差別
網域版	Deluxe SSL (OV)	1~3 年	網域與公司擁有者	標準保護
商業版	Extended Validation (EV)	1~3 年	網域與公司擁有者會計資料	於網址列有綠色標記，保證網站的安全、可用、擁有者合法狀態。

EV 版範例：https://developer.apple.com(查詢日期:2016/10/1)

網域版範例：https://www.owasp.org/(查詢時間：2016/10/1)

單機版範例：https://ep.land.nat.gov.tw/(查詢時間：2016/10/1)

台灣 GCA 在 iPhone 瀏覽器上不被信任

安全開發生命週期：開發實作階段

不安全程式碼範例：主機：Server

OpenSSL 1.01f(含以前)版本

2048 位元以下、逾期、失效、根憑證失效、掛失等憑證。

安全程式碼範例：主機：Server

OpenSSL 1.01g(含之後) 版本

2048 位元(含)以上，合法有效憑證。

行動應用 App 基本資安規範：N/A

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：N/A

Server-11:參考 CPM-01~CPM-08 實作會談管理(Session Management)伺服器端設定。

說明：

網頁伺服器的 Session 資料常被使用在瀏覽器通訊中，但卻是有可能被破解。保持伺服器軟體處於最新的安全更新。確保 Session cookie 的長度是足夠的，建議提高 Session 的安全等級。

依照建議：CPM-01:行動應用 App 實作 CPG-02 TLS 連線，需使用編碼長度為 128 位元(含)以上之交談識別碼

安全開發生命週期：開發實作階段

不安全程式碼範例：網站主機：.NET

.NET 疑慮：SessionID 預設為 120 位元

ASP.NET 的 SessionID 是由 24 個字元構成，每個字元是由小寫 a 到 z，加上 0 到 5，即 32 種組合，實際有效的位元數是 2 的 5 次方。總共是 24*5=120 位元

安全程式碼範例：網站主機：.NET C#

實作 SessionIDManager：128 位元之 SessionID

```
//
// ISessionIDManager.CreateSessionID
//
public string CreateSessionID(HttpContext context)
{
  //Guid 的值是 128 位元
  return Guid.NewGuid().ToString();
}
```

行動應用 App 基本資安規範：N/A

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：N/A

Server-12:使用網路區隔及網路存取控制措施，保護不應被外部存取的內部資源。

說明：

內部資源不能被外部存取，如管理功能網站。

也應注意有些安裝的路徑在安裝之後必須被刪除或移動到其他目錄。

安全開發生命週期：部署維運階段

不安全程式碼範例：

網站主機：將 admin 置於開放網站

例如：Open Cartg 管理目錄 \admin

例如：osCommerce 管理目錄 catalog\admin

網站主機：未將安裝目錄移除。

例如：Open Cart 安裝目錄 \install

例如：osCommerce 安裝目錄 catalog\install

安全程式碼範例：N/A

行動應用 App 基本資安規範：N/A

行動應用 App 基本資安檢測基準：N/A

各國開發要點參考：N/A

參考來源：post installation - osCommerce (查詢時間：2016/10/1)

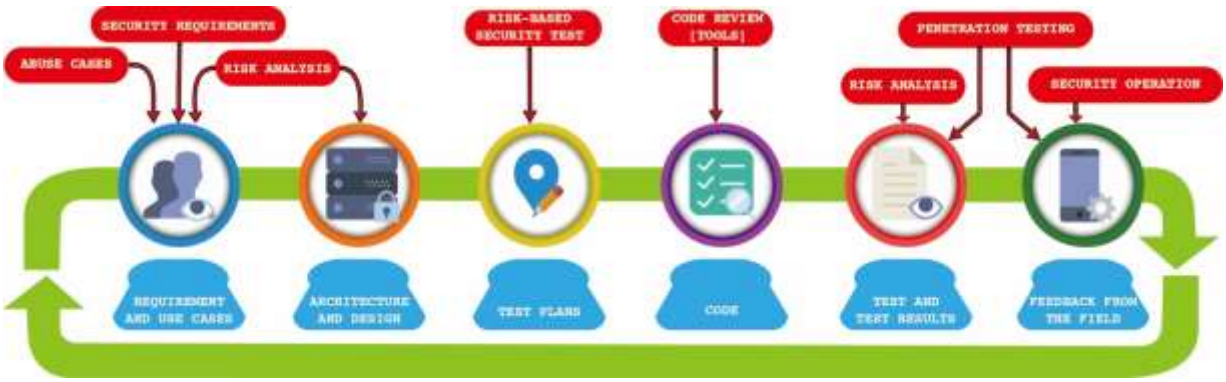
4. 行動應用 App 安全開發生命週期(SSDLC)

「安全是被設計，而不是被檢查(駭)出來」，是一個必須要深植在各行各業的工程、技術人員腦海的基本觀念，這當然也包括行動應用 App 企劃、專案管理、設計、開發及測試的相關從業人員。

行動應用 App 從被企劃出來，到正式上架的時程一般來說都很短，或者僅是一個資訊系統的某一項使用者介面，而行動應用開發人員，大部分為小型工作室或個人，沒有大型開發單位有完整的專案經理、資訊安全人員、系統架構師、系統分析人員、系統設計人員、程式開發人員、系統測試、軟體品質管理人員、系統管理員及資料庫管理員般細密的分工，以採取傳統軟體安全開發流程。

即使像 Google、facebook 如此軟體巨擘，為了應付需求快速改變、加速生產，也多採敏捷式開發(Agile software development)，相對於「非敏捷」開發方法論，更強調程式設計師團隊與業務專家之間的緊密協作、面對面的溝通(認為比書面的文檔更有效)、頻繁交付新的軟體版本、緊湊而自我組織型的團隊、能夠很好地適應需求變化的程式碼編寫和團隊組織方法，也更注重軟體開發過程中人的作用。

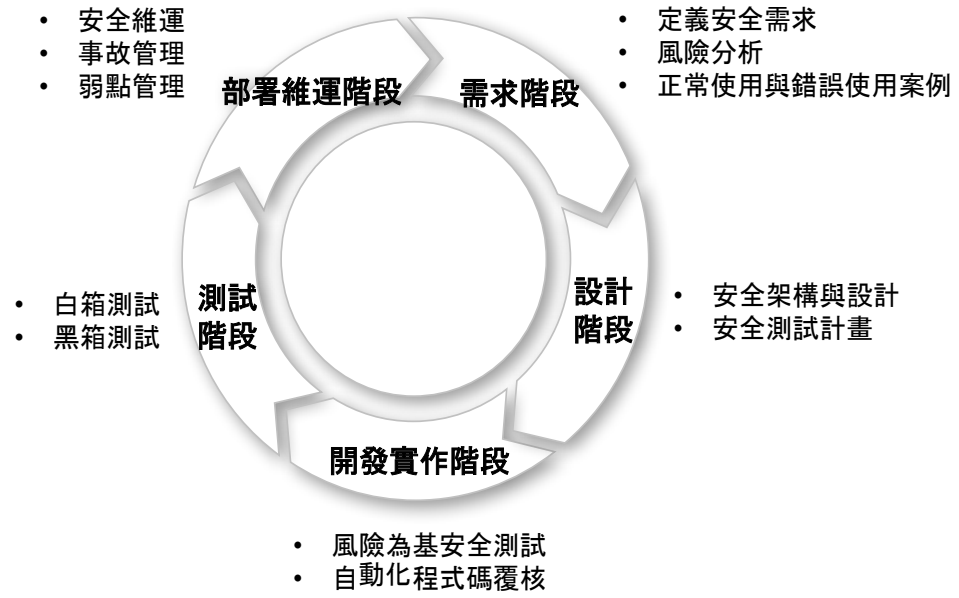
但在快速開發過程，軟體品質與行動應用 App 的安全性，仍然是不可或缺元素。加強 App 安全，在傳統軟體安全生命週期(Software Development Lifecycle, SDLC)上架構安全活動的安全開發生命週期(Secure Software Development Lifecycle, SSDLC)，常見 SSDLC 方法論有 Cigital 公司的 Touchpoint、Microsoft 公司的 Security Development Lifecycle(SDL)及開源軟體安全計畫(OWASP)的 Security Assurance Maturity Model(SAMM)，此三項方法論比較摘述請詳見附件 1，本指引以 Cigital 公司 Touchpoint 的安全方法論為例：



資料來源：www.cigital.com

圖22 Cigital Touchpoint 方法論

將安全活動分為「需求階段」、「設計階段」、「開發實作階段」、「測試階段」及「部署維運階段」，各階段主要活動詳見圖 23 所示及說明如下：



資料來源：本計畫整理，參考 Software Security Building Security In，Gary McGraw,Cigital CTO（查詢時間：2016/10/1)

圖23 安全軟體生命週期

就上圖呈現在 SSDLC 在傳統 SDLC 各階段，增加不少資訊安全活動，在就一次性專案角度來看 SSDLC 似乎讓專案成本大幅提高，但如果以多次專案及整體因減少軟體漏洞導致安全事故後所需要修補的損失成本來看，反而會大幅降低總體持有成本(Total cost ownership, TCO)。建議可以先就 SSDLC 在傳統開發流程的安全活動，再依敏捷式開發來調整簡化，保持靈活彈性其兼顧安全需求。

資料來源：本計畫整理，

參考 Software Security Building Security In，Gary McGraw,Cigital CTO & http://www.screenmedia.co.uk/blog/2014/08/what-is-agile-development-a-brief-introduction/

圖24 敏捷開發示意圖

敏捷式開發可以每個衝刺(Sprint)的產出物進行自動化原始碼檢測工具測試，無需產生人工編製報告，依工具產生結果決定是否再回頭修改程式碼，直到通過安全測試完成產品。

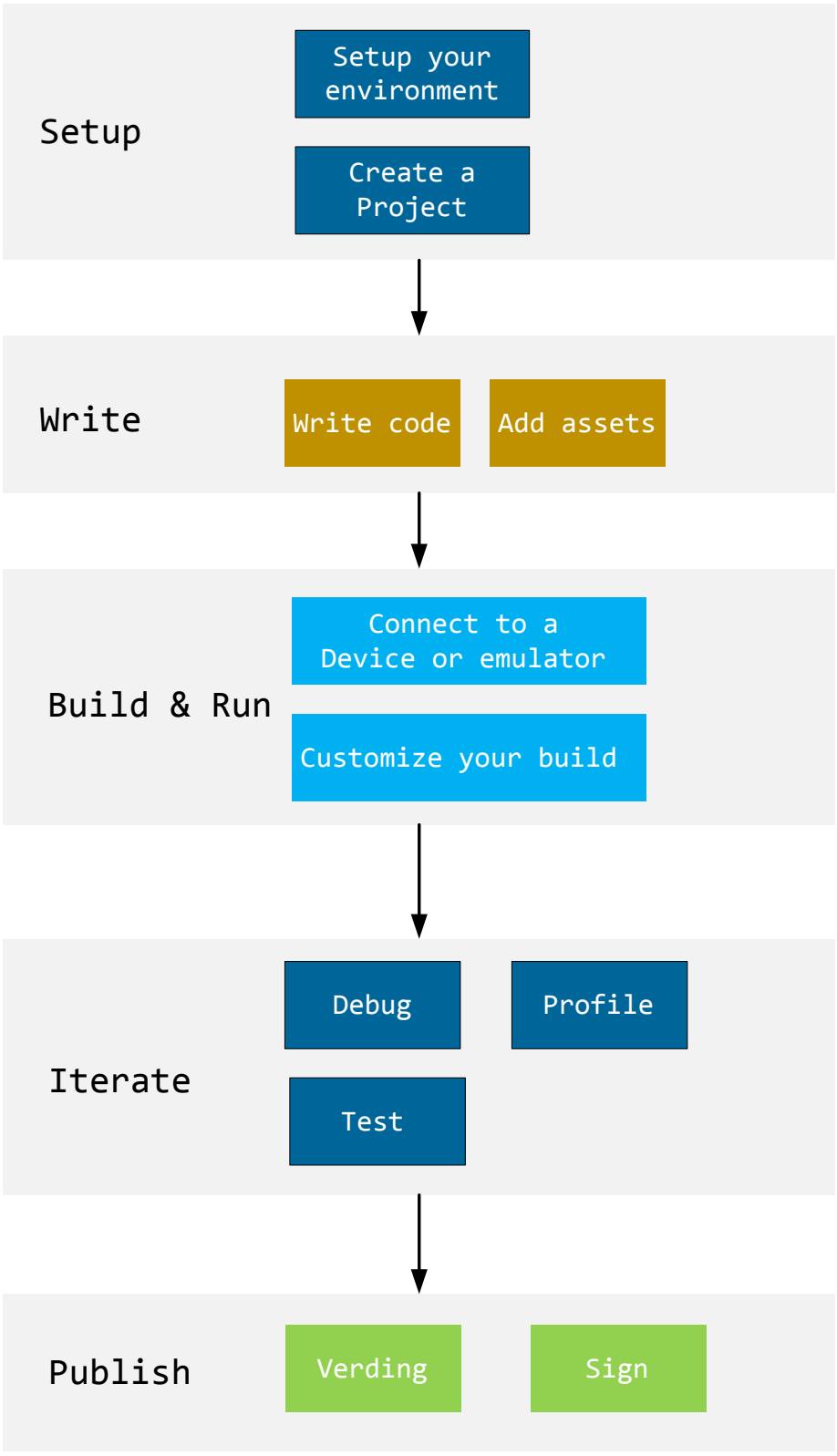
4.1.準備階段

在行動應用 App 專案正式開始之前，不論中大開發團隊、小型工作室、個人開發人員除了準備開發環境外(參考本指引第 2.1 與 2.2 章節)，需要準備至少包括下列事項：

- 建立安全開發流程。
- 安全開發教育訓練。
- 測試第三方 API/函式庫。
- 建立原始碼安全控管機制。

4.1.1. 建立安全開發流程

不論開發團隊大小，採取何種開發方法論，建立適當開發流程是必需的，中大型的開發團隊，若採用傳統 SDLC 可能會有完整「應用系統開發管理規範」，包含需求單、系統分析/設計文件、程式規格書、測試報告及上線/過版申請單等文件；採用敏捷式開發可能運用 Trello、Jira 等專案或問題追蹤工具，建立專案團隊溝通及確保軟體問題被有效解決。個人開發人員則可參考如 Android 建議基本流程，詳見圖 25 所示。



資料來源：Android Developer

圖25 Android Developer Workflow Basics

4.1.2. 安全開發教育訓練

在準備階段最重要是接受安全開發教育訓練，培養資訊安全意識，對行動應用 App 常見及近期發現的資安威脅保持警覺，並學習行動應用安全開發實務，與行動應用 App 開發相關人員(包括 PM、IS、SA、SD、PG 及 QA)，每年至少接受 1 次以上訓練，或持續參加外部相關訓練或研討會，其訓練主題應該包含以下範圍：

- 安全的軟體設計，包含如何減少攻擊面(Attack Surface)、縱深防禦、最小權限法則與預設安全性(參考本指引第 2.3 節行動應用 App 資安風險議題)。
- 安全的程式開發，包含緩衝區溢位、SQL 資料隱碼攻擊、跨站腳本攻擊與避免脆弱的密碼學應用。(參考本指引第 3 章行動應用 App 安全開發最佳實務)
- 安全的軟體測試，包含安全測試與功能性測試的差異、風險評鑑與安全測試方法。(參考本指引第 5 章行動應用 App 資安檢測實務)
- 行動應用平台最新安全機制(參考本指引第 2.1 節)。
- 我國最新行動應用 App 基本資安規範(參考本指引第 2.5 節)，我國行動應用 App 基本資安規範第 4 章技術要求與 SSDLC 對應關係，詳見圖 26 所示。

技術要求 / SSDLC	4.1. 行動應用程式資訊安全技術要求事項					4.2. 伺服端資訊安全技術要求事項
	4.1.1. 行動應用程式發布安全	4.1.2. 敏感性資料保護	4.1.3. 付費資源控管安全	4.1.4. 身分認證、授權與連線管理安全	4.1.5. 行動應用程式碼安全	本規範旨在針對行動應用程式安全提出基本資訊安全要求，如行動應用程式涉及伺服端之資訊安全需求，建議應由業者自我宣告或切結其伺服端資訊安全防護與管理措施，或對於其伺服端服務之資訊安全防護與管理，出具第三方檢測通過證明。
A.需求階段	N/A	4.1.2.1. 敏感性資料蒐集 4.1.2.2. 敏感性資料利用 4.1.2.3. 敏感性資料儲存 4.1.2.4. 敏感性資料傳輸 4.1.2.5. 敏感性資料分享 4.1.2.6. 敏感性資料刪除	4.1.3.1. 付費資源使用 4.1.3.2. 付費資源控管	N/A	N/A	
B.設計階段	N/A			4.1.4.1. 使用者身分認證與授權 4.1.4.2. 連線管理機制	4.1.5.1. 防範惡意程式碼與避免資訊安全漏洞 4.1.5.2. 行動應用程式完整性 4.1.5.4. 使用者輸入驗證	
C.開發實作階段	N/A					
D.測試階段	N/A					
E.部署維運階段	4.1.1.1. 行動應用程式發布 4.1.1.2. 行動應用程式更新 4.1.1.3. 行動應用程式安全性問題回報	N/A	N/A	N/A	4.1.5.3. 函式庫引用安全	

資料來源：本計畫整理

圖26 我國行動應用 App 基本資安規範第 4 章技術要求與 SSDLC 對應關係

4.1.3. 測試第三方 API/函式庫

開發人員為增加行動應用 App 功能或開發效率會引用第三方(開源或商業授權)API/函式庫，引用前需要先經安全測試，確認沒有已知漏洞或有不明背景傳輸行為。開發團隊必須完成第三方 API/函式庫安全測試後，才能納入引用共享資源，避免為臨時性需求，引用不安全第三方 API/函式庫，造成資安漏洞，像 2014 年被發現 OpenSSL heart breed 漏洞，影響層面就非常廣泛。

4.1.4. 建立安全程式碼控管機制

原始程式碼對開發人員而言，屬於付出心血的智慧財產，對攻擊者是入侵竊取機敏資料的珍寶，經安全檢測的原始碼或 API 可以重複引用，降低行動應用 App 漏洞發生機率，建立安全原始程式碼控管機制，如權限控管、簽入簽出流程及版本控管。可以透過版本控管工具及變更管理機制確保不會誤用不安全程式碼。

完成前述基本準備工作，就可以進入行動應用 App 安全開發生命週期，藉由各階段安全活動來降低安全風險

4.2.需求階段

本階段應由 PM 及 SA 識別與定義行動應用 App 安全需求，行動應用 App 的需求單位或業主在企劃需求時，通常偏重提出功能性及效能性需求，如開發線上購物行動應用 App 的需求，可能會是商品要如何在行動智慧裝置上被呈現、會員資料蒐集、購物車操作流程及與社群軟體整合等需求，安全性需求可能只會應因法規要求或是資料正確性提出基本的安全需求，就 PM/SA/IS 人員除了功能性的需求蒐集外，也需要進行下列安全活動(安全接觸點)：

●定義安全需求。

●風險分析。

●定義正常使用與錯誤使用案例。

4.2.1. 定義安全需求

PM/SA/IS 人員就行動應用 App 的需求單位或業主提出安全需求，至少確定認下列事項：

●行動應用 App 是否有蒐集、處理及利用個人資料保護法第二條定義的個人資料或其他敏感資料?

- 行動應用 App 是否有來自產業的資料安全規範，如支付卡產業資料安全標準(PCI DSS)、銀行公會金融機構辦理電子銀行業務安全控管作業基準等。
- 行動應用 App 是否有上架當地國家資料隱私相關法規，如歐盟隱私資料保護指令及日本個人情報保護法等。
- 行動應用 App 敏感性資料生命週期(蒐集、利用、儲存、傳輸、分享及刪除)的保護需求。
- 行動應用 App 的敏感性資料是否有接收自或傳送至第三方的需求?
- 行動應用 App 是否涉及付費或金流機制?
- 行動應用 App 的功能需求對比「行動應用App基本資安規範」的安全分類屬於那一類：
  - 第一類：純功能性。
  - 第二類：具認證功能與連網行為。
  - 第三類：具交易功能(包括認證功能及連網行為)。
- 行動應用 App 使用對象是一般不特定消費者還是企業內員工。

從以上的法規面、營運面相關問題去引導業主或是需求單位識別出安全性需求，確認後納入需求清單。

4.2.2. 風險分析

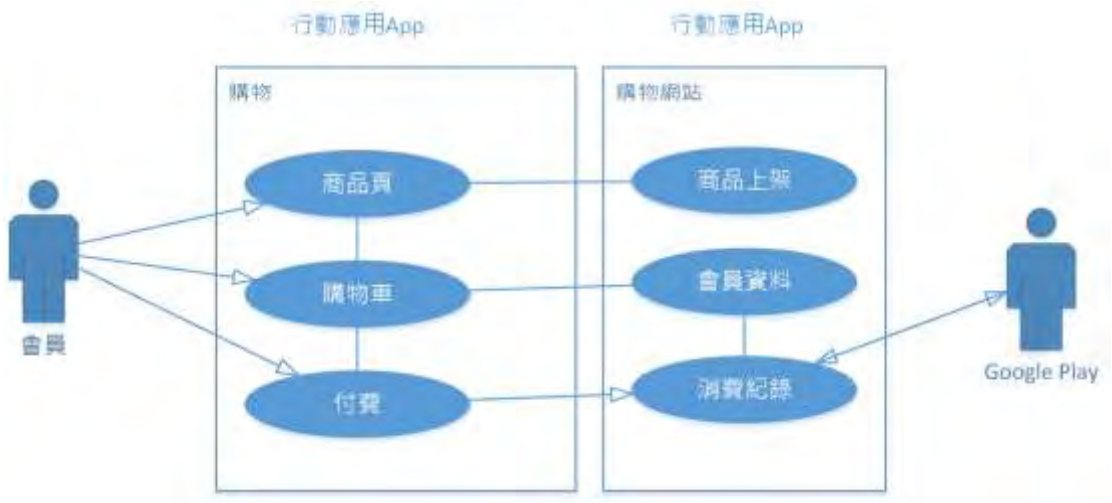
由 PM/SA/IS 與業主或需求單位共同討論行動應用 App 的使用情境，考量以攻擊者的角度，識別可能影響行動應用 App 與後端系統的威脅並進行評估。基於對既有及預計架構與實作方法的了解，識別與評估潛在威脅及可能產生弱點後，僅做定性的風險分析，以風險高低的排序選用適當的控制

措施，以利快速研擬行動應用 App 開發專案的安全需求及所需資源投入重點項目。

4.2.3. 定義正常使用與錯誤使用案例

建立使用案例的目的在於將業主或需求單位未於訪談時表達的潛在需求具體化，並對後續開發實作、測試對應需求提供參考模型，以下為各 SDLC 階段中使用案例的相關活動：

- 需求(Requirements)：使用案例(Use cases)模型就是需求分析的結果，可以從使用案例規格中找出專案詞彙和建立領域或概念模型。
- 分析與設計(Analysis and Design)：使用案例可以轉化成分析與設計模型，用以描述使用案例如何在分析和設計模型中實作，表示此使用案例進入分析或設計狀態。
- 實作(Implementation)：因為使用案例模型是設計模型的基礎，實作的程式碼就是在執行特定的使用案例，當撰寫程式碼執行每一個使用案例後，就表示此使用案例進入實作狀態。
- 測試(Testing)：可以從使用案例模型建立測試案例和測試程序，軟體系統可以借由執行每一個使用案例來進行測試，通過測試就表示使用案例進入驗證狀態。正常使用案例示意圖詳見圖 27 所示。

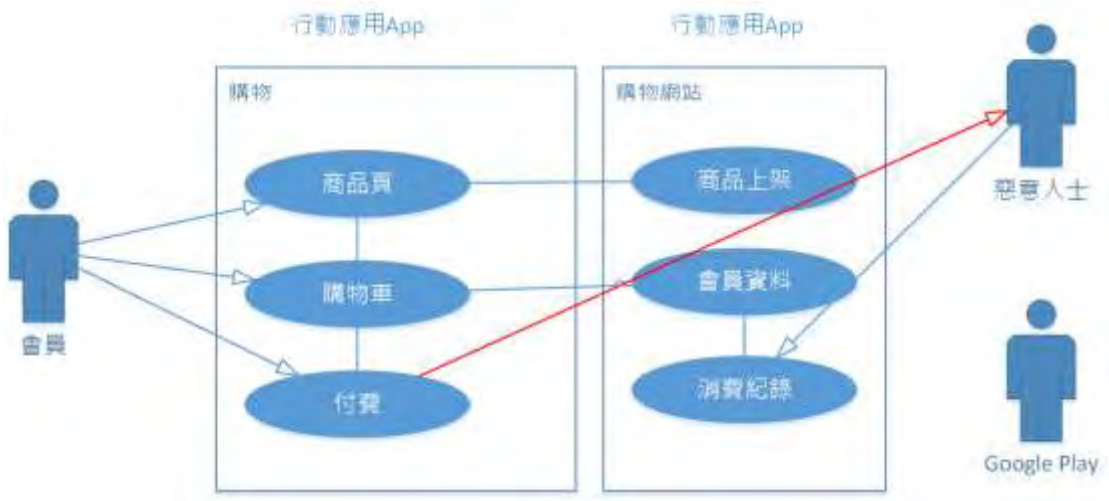


資料來源：本計畫整理

圖27 正常使用案例示意圖

從使用案例可發展出誤用案例，透過發展負面的使用情境來幫助識別安全需求，負面的使用情境是系統中意料之外的行為，為系統擁有者不希望發生的情況。藉由採用攻擊者的角度，誤用案例反轉了原先使用案例中正常使用者的視角，提供發覺「外部威脅」如何在軟體或系統中發生之機會。

誤用案例的建模方式相似於使用案例，兩者的繪製方式相同，但是誤用案例記錄的是刻意或意外的狀況下，誤用者如何以非原本預期的方式與軟體進行互動。誤用案例示意圖詳如圖 28 所示，惡意人士攔截金流，並回覆已完成付費。



資料來源：本計畫整理

圖28 誤用案例示意圖

為使開發人員(擔任 PM/SA/IS 角色)執行需求階段需要從事的安全活動有所依循，本指引整理「需求階段安全檢核表」詳見附件 2。

4.3.設計階段

安全與隱私防護功能應及早於系統設計初期納入，以避免於開發後期再行納入所導致之成本大幅增加，系統設計(SD)人員實際執行時應針對相關功能架構描述安全實作方法，如有必要需與資料庫 DBA 及負責伺服器維運人員(MIS)討論架構設計可行性與安全性，包括威脅建模、限制非必要服務、最小權限法則及強化縱深防禦等採取手段，可依據行動應用 App 基本資安檢測基準建立查核點。在設計階段，需要進行的安全活動為：

●安全架構與設計。

●安全測試計畫。

4.3.1. 安全架構與設計

安全架構師或系統設計人員依需求階段產生的「安全需求」、「風險分析結果」及正常與誤用案例來設計系統架構。初次設計架構優先考量運用

Android 或 iOS 行動應用平台提供軟、硬體內建安全功能，如 Android 的應用系統沙箱、強化 Linux 內核及官方提供安全 API/函式庫；iOS 的安全啟動鏈、Keychain、ATS 及檔案加密系統。其次參考 Android 與 iOS 的安全開發指南建議及本指引第 3 章安全開發最佳實務進行 App 本地端及伺服器間安全設計，產生系統架構與設計文件。

設計時需考量系統架構是否可以預防、偵測或緩解惡意攻擊，如避免使用硬體 ID 作為認證身分用的 ID 或如需運用增加額外加密措施以保護，另需考量實作控制措施的可行性、成本及對系統效能衝擊。

對敏感性資料的保護，應以整個資料生命週期進行考量，選用符合法規、行業規範的控制措施。

安全架構與設計以整體安全考量，不單僅為單一專案考量，以便未來類似專案套用安全架構可套用，無需完全從零開始，並做好行動應用 App 相關安全性更新與新舊版 App 上架管理等機制。

系統架構與安全設計，至少需包括：

●系統架構圖。

●儲存區分配規劃。

●資料儲存安全設計。

●資料傳輸安全設計。

●安全界面設計。

●資料消除設計。

●權限設計。

●App 間安全通訊設計。

4.3.2. 安全測試計畫

在完成系統架構與安全設計後，在轉移至開發實作階段前，應依需求階段安全需求與架構設計，完成以風險為基礎的對應安全測試計畫，以利在開發實作完成後於測試階段進行安全需求驗證。

測試計畫研擬應包含如下事項：

●測試目標。

●測試策略

– 風險分析。

– 測試範圍。

– 測試方法。

– 測試環境。

●測試程序及接受標準。

●資源與時程規劃。

●測試報告格式。

●應變措施。

為使開發人員(擔任系統設計角色)執行設計階段需要從事的安全活動有所依循，本指引整理「設計階段安全檢核表」詳見附件 3。

4.4.開發實作階段

包括建立經過認可之安全開發工具清單，對於所有使用之函式庫應審查其安全歷史，並研擬安全之替代方案，程式開發人員(PG)需依行動應用 App 安全開發實務(第 3 章詳述)撰寫安全程式碼並實施單元測試，可依據行動應用 App 基本資安檢測基準建立查核點，同時本階段應執行程式碼靜態分

析，可使用靜態自動分析工具或由主管或不同開發人員實施必要之人工審核。

4.4.1. 瀑布式 SSDLC 與敏捷式開發

本階段可視開發人員使用傳統瀑布式(Waterfall)SDLC 還是使用類似敏捷式開發方式在以每個衝刺(Sprint)寫程式碼及測試之間，不斷反覆循環(iteration)。敏捷開發是一種以人為核心、更迭、循序漸進的開發方法。在敏捷開發中，軟體專案的建構被切分成多個子項目，各個子專案的成果都經過測試，具備整合和可運行的特性。換言之，就是把一個大專案分為多個相互聯繫，但也可獨立運行的小專案，並分別完成，在此過程中軟體一直處於可使用狀態。

由麻省理工學院史隆管理學院評論(MIT Sloan Management Review)所刊載的一篇為時 2 年對成功軟體開發專案的研究報告，報告指出了軟體開發專案獲得成功的共同因素，排在首位的是反覆循環開發方法，而不是瀑布式開發方法。其因素為：

- 至少每天把新程式碼合併到整個系統，並且通過測試，對設計變更做出快速反應。
- 開發團隊具備運作多個產品的工作經驗。
- 很早就致力於建構和提供聚合的架構。

由上可知開發團隊需要良好的程式編寫實務經驗，其中當然包括安全開發實務，所以在實行行動應用 App 敏捷式開發需要熟悉本指引第 3 章建議的安全實務，再輔以靜態自動分析工具，才能維持敏捷的特性。

4.4.2. 行動應用 App 之 DevOps 開發流程

DevOps 是一種重視「軟體開發人員(Dev)」和「IT 運維技術人員(Ops)」之間溝通合作的文化、運動或慣例。透過自動化「軟體交付」和「架構變

更」的流程，來使得建構、測試、發布軟體能夠更加地快捷、頻繁和可靠。關於 DevOps 相關資訊可參考 ithome 的專欄(查詢時間：2016/10/1)

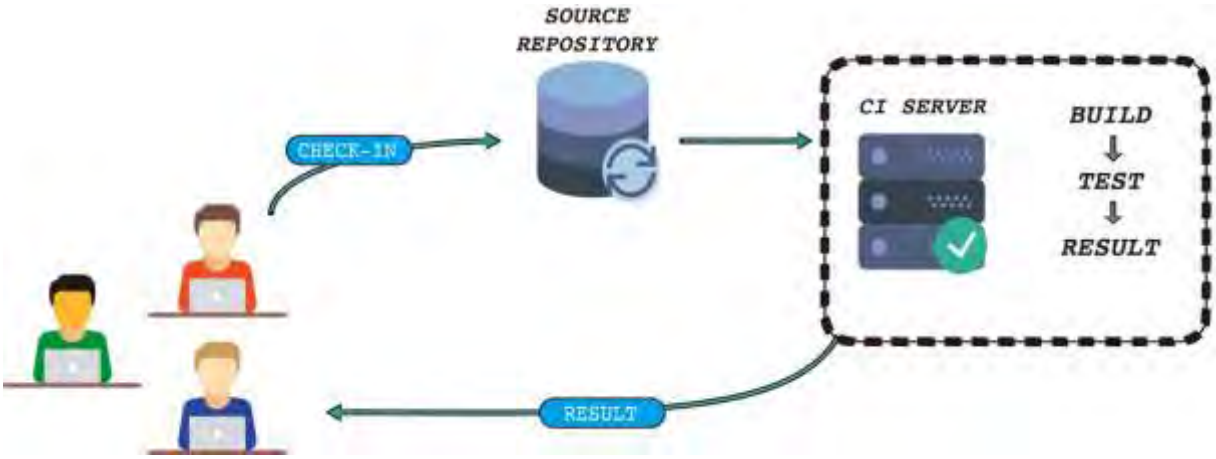
DevOps 為近年開始在小型開發團隊流行敏捷式開發方法之一，亦有大型企業，如 Yahoo、IBM、Red Hat 及微軟等科技業外，梅西百貨與迪士尼也紛紛擁抱 DevOps，DevOps 是用來打破開發團隊與營運團隊高牆，就傳統的 SDLC 而言，基於安全考量，區分為開發團隊與維運團隊，開發團隊不能接觸正式營運環境與資料，程式碼開發完成後需經完整系統整合測試(System Integration Testing, SIT)及使用者接受測試(User Acceptance Testing, UAT)，並經過弱點掃描或滲透測試，再由維運團隊建構執行碼，在系統更新對業務衝擊最小時上版更新，如發生問題需要退回前一版本，在此過程為符合安控需求，所有應用變更過程需保留書面紀錄，導致整體開發及上線週期相對冗長，較不適合行動應用 App 開發週期偏短的特性。

以行動應用 App 開發而言，因本身不涉及組織 IT 基礎架構穩定度問題，可以透過 DevOps 進行持續整合(Continuous Integration, CI)工具可以從版本控制系統一路串接到測試、布建和發布的 Dev 全程自動化，做到持續交付與持續部署。

參考 Suzie Prince on, 2016/2/11 發表 The Product Managers' Guide to Continuous Delivery and DevOps，文中對「持續整合(Continuous Integration)」、「持續交付(Continuous Delivery)」和「持續部署(Continuous Deployment)」，將這 3 個概念說明如下：

●持續整合

持續整合強調開發人員上傳了新程式碼之後，立刻進行建構、單元測試。根據測試結果，可以確定新程式碼和原有程式碼能否正確地整合在一起。

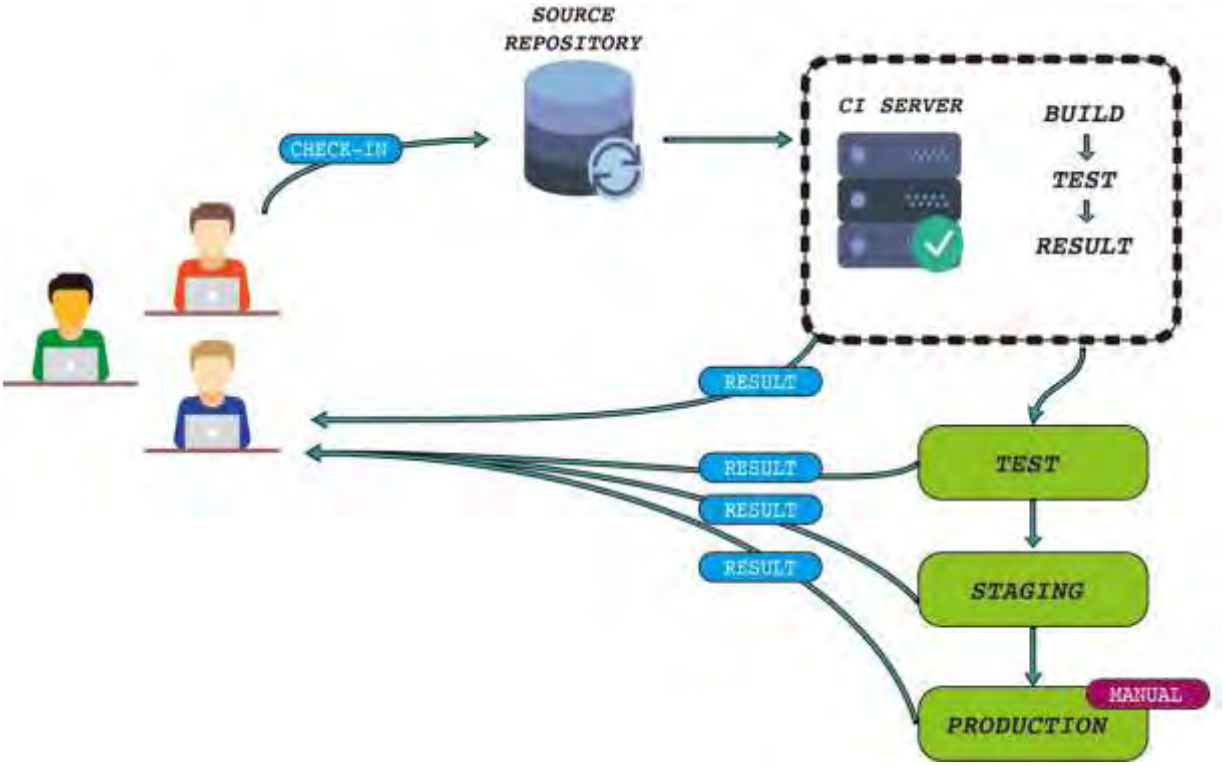


資料來源： The Product Managers, Guide to Continuous Delivery and DevOps (查詢時間：2016/10/1)

圖29 DevOps 持續整合示意圖

●持續交付

持續交付在持續整合的基礎上，將整合後的程式碼部署到更貼近真實運行環境的「類正式環境」(production-like environments)中。比如，於完成單元測試後，可以把程式碼部署到連接資料庫的 Staging 環境中更多的測試。如果程式碼沒有問題，可以繼續手動部署到正式環境中。

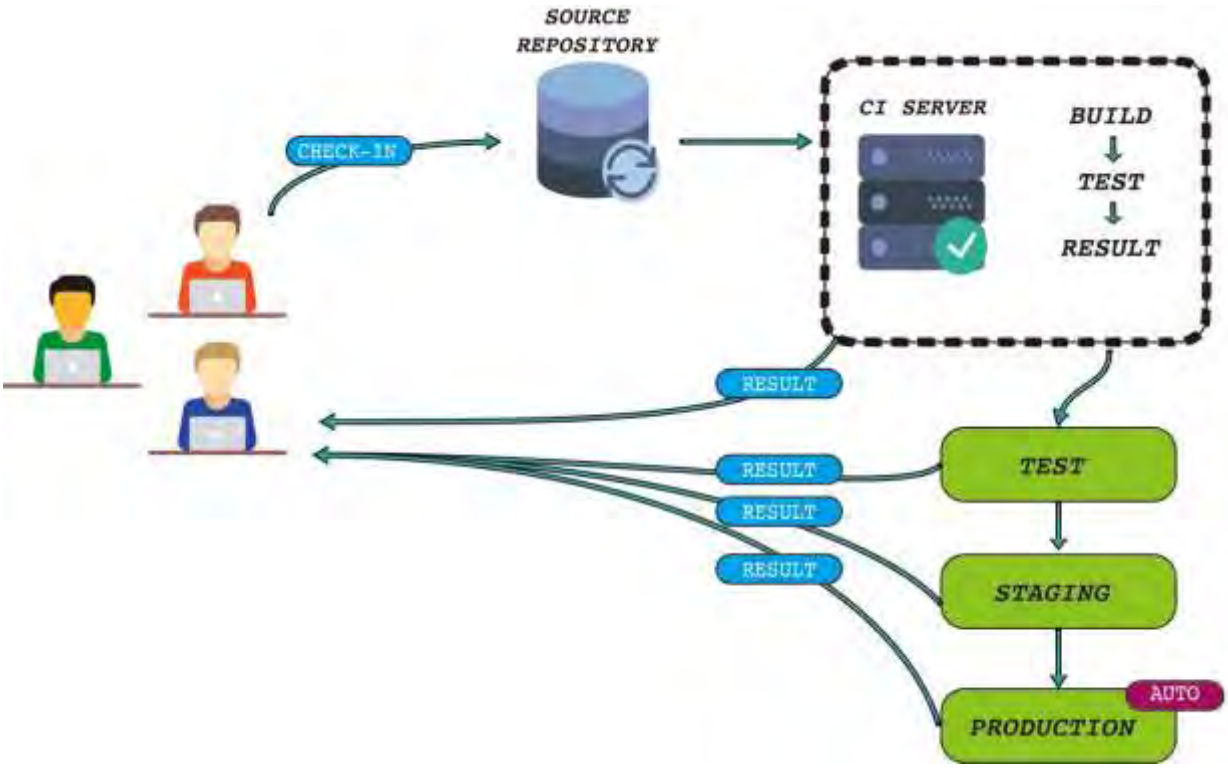


資料來源：[The Product Managers, Guide to Continuous Delivery and DevOps(查詢時間：2016/10/1)]

圖30 DevOps 持續交付示意圖

●持續部署

持續部署則是在持續交付的基礎上，把部署到正式環境的過程自動化。



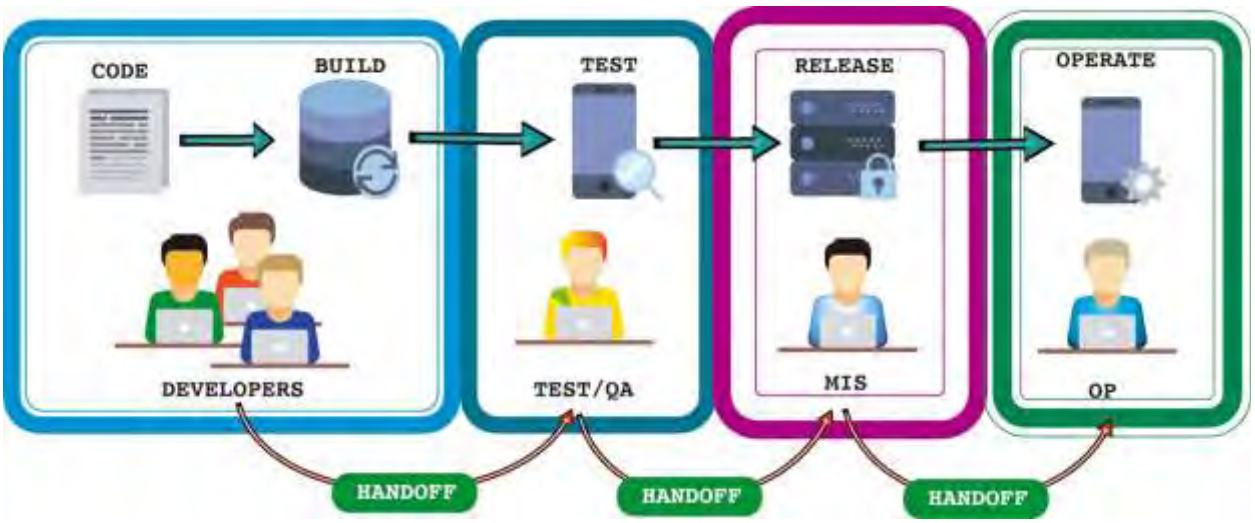
資料來源：The Product Managers,Guide to Continuous Delivery and DevOps（查詢時間：2016/10/1)

圖31 DevOps 持續部署示意圖

●傳統開發/測試/上線/維運與 DevOps 的對比流程說明如下：

– 傳統開發/測試/上線/維運流程

開發人員(Developer)完成程式碼撰寫、單元測試及建構(Build)後，就移交(Handover)給測試人員及使用者完成 SIT 與 UAT 後，再轉移給上線人員上版至正式環境交由維運人員(系統管理員、SP、OP 或 MIS)，經過 3 次以上的移交給不同角色，在溝通及切換不同 IT 環境的相對 DevOps 會較為複雜及時程冗長。示意圖詳見圖 32 所示。

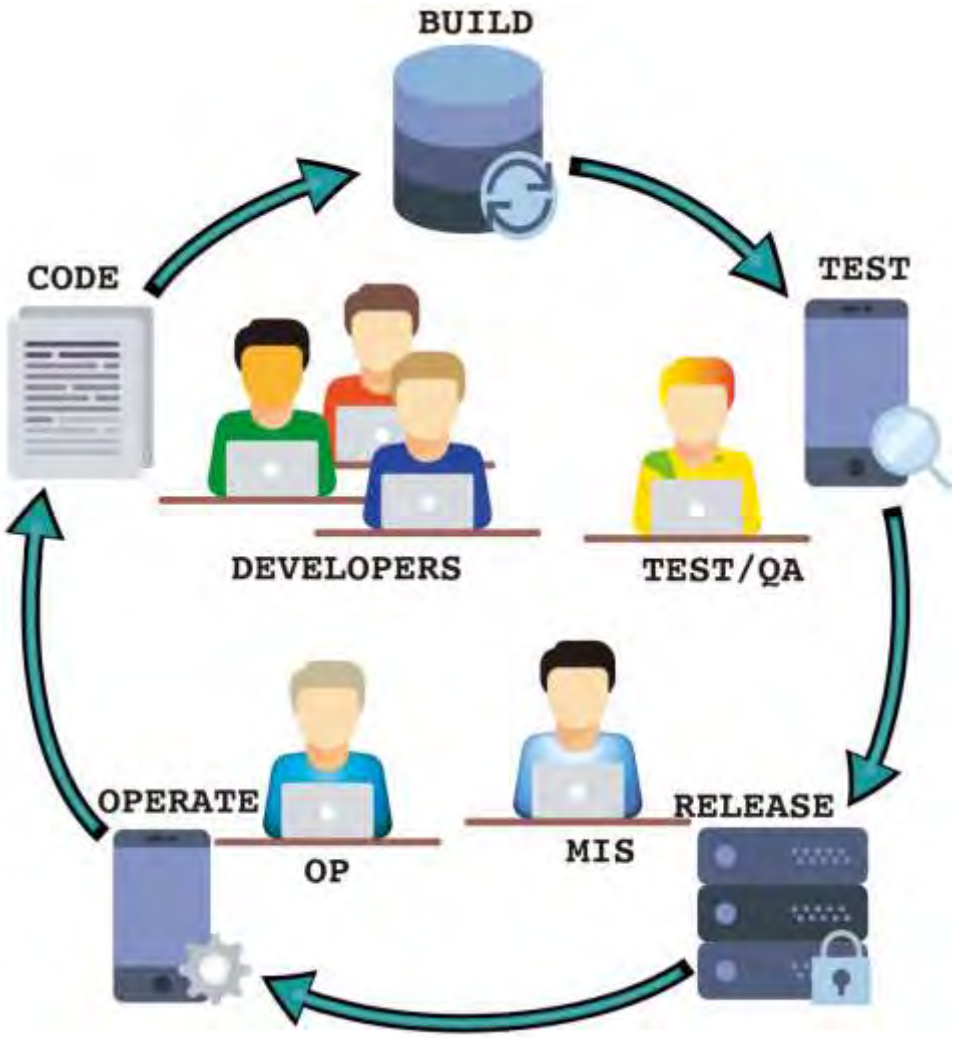


資料來源：The Product Managers' Guide to Continuous Delivery and DevOps （查詢時間：2016/10/1)

圖32 傳統開發/測試/上線/維運流程示意圖

– DevOps 的開發/測試/上線/維運流程

DevOps 的開發/測試/上線/維運流程與傳統流程最大的差異就是各角色以專案為單位整合成為單一團隊，持續整合、持續交付及持續部署不間斷，藉由自動化工具與 DevOps 運作團隊文化，加速 App 產品化效率。示意圖詳見圖 33 所示。



資料來源：The Product Managers' Guide to Continuous Delivery and DevOps （查詢時間：2016/10/1)

圖33 DevOps 開發/測試/上線/維運流程示意圖

以本指引的安全開發實務，於開發實作階段以單元測試自動化較為可行，先做到持續整合，再來是以 Use case 做到系統整合測試自動化的持續交付，無法自動化人工檢測項目，於測試階段進行一次性測試，所以現階段做到持續交付是比較可行，如要做到持續部署，需要要改變舊有系統架構及部署流程，在實作上不易自動化成功。

為使開發人員執行開發實作階段需要從事的安全活動有所依循，本指引整理「開發實作階段安全檢核表」詳見附件 4。

4.5.測試階段

本章節說明 App 測試階段之檢測目標、檢測流程與檢測方法與測試檢核表，透過 App 測試階段之各項說明，可以了解同樣屬於「軟體」的 App 該如何透過檢測而達到一定的基本安全品質。在前一節提到傳統 SDLC 開發模式下 App 的測試階段中，可預期以人工方式在開發環境、測試環境及發布轉移的過程中會遭遇到任何可能的問題與狀況，因此「持續測試」與「持續部署」將會時常發生而可能造成開發、測試與維運團隊間的負擔與失誤。

此時可以考量一個比較有效的溝通協同合作的開發方法，如 DevOps，應較適合 App 在人員密集溝通與合作、運用整合性開發環境、自動化測試工具、基礎設施調整配置優化及持續提高自動化程度，產出效率等得到有效的協作溝通運作綜效。

藉此可以確立開發測試範圍、強化發布協調，並以全自動化測試為目標，以將本指引之開發、測試實務相關建議融入日常運作流程，獲得一個適合團隊開發的整合環境。

團隊可透過此一整合環境，在高效產出的目標為前提上，仍需兼顧品質與安全的共識，並可作為協助測試人員或開發人員蒐集損毀資料(Crash Report)、使用者意見(Feedback)、發布管理(Distribution)及分析(Analytics)等，並以本指引之資安檢測基準為基礎作為持續測試部署的整合開發流程。

在導入 DevOps 之後，隨著成熟度提高，開發與測試階段界限會逐漸模糊，本指引建議應優化測試流程，初步區分可自動化測試項目的人工測試項目，可分流同步進行以縮短測試階段時程，並提高測試有效性。

4.5.1. App 檢測目標

App 檢測主要目標在於 App 品質的分析與檢驗，而 App 軟體測試工程中強調驗證與確認軟體在產出的設計規格與符合需求，在 ISO/IEC 12207 標準中描述驗證在於「檢查軟體產品並提出客觀證據，以證實達成訂定的規格要求。」與「判斷某發展階段的軟體產品，達成前項發展階段訂定的需求或限制。」，在確認部分則是「檢查軟體產品並提出客觀證據，以證實達成某一特定預期功用的需求」與「確定依據需求規格製作的最終軟體產品，是否滿足特定使用目的。」；在 CMMI 相關指引中描述驗證的目的在於「確保工作產品符合其指定需求的規格」，而確認的目的則是「在展示最終軟體產品或產品元件在需求環境中，實現客戶需要的產品。」。

此外，學者 Barry Boehm 則強調驗證是否用正確的方法與步驟建構產品以及確認建構的軟體系統是否滿足使用者的真正需求。因此在驗證及確認之工作中必須確保產出之 App 能符合客戶之功能或非功能性要求，而其中安全規格已是當前政府與使用者重視的重要要求。

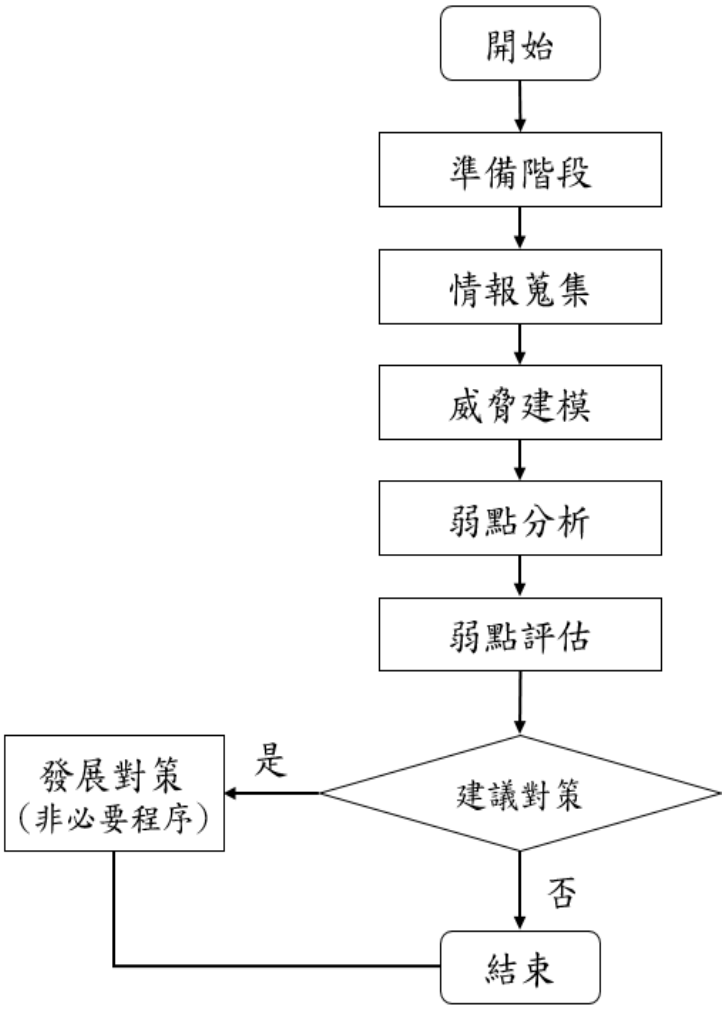
本指引以發展提供開發人員自主 App 檢測指引為目標，並考量開發人員在資訊安全之能力水準，且以符合經濟部工業局公告之「行動應用 App 基本檢測基準 V3.0」中之規範為主要基礎。本指引提出 App 檢測指引與檢測工具說明等方案成果，藉此可針對 App 進行完善檢測並提供保障客戶與相關使用者資料或相關安全需求之產品服務。

4.5.2. App 檢測流程

本指引參考 OWASP 所提出針對行動 App 的安全測試指引，採用其基本核心測試流程為基礎，提出本指引整合最新適合我國「行動應用 App 基本檢測基準 V3.0」與開發人員使用之 App 檢測流程，作為本指引後續方法與工具具體可實施之發展基礎。透過此流程之階段程序可指引開發人員擬定

App 自我檢測計畫、執行檢測計畫、判讀檢測報告以及取得相關建議方向後進行 App 安全性之改善。

本指引參考採用之基本 App 核心檢測流程(詳見圖 34 所示)，透過 App 資料擷取、威脅建模及弱點分析等流程，最後以指引說明或工具檢測等方式，獲得自我檢測結果與建議。



資料來源：The OWASP Foundation

圖34 基本 App 檢測核心流程

基本 App 檢測核心流程包含不同階段需進行之內容，相關重點內容說明如下：

4.5.2.1. 準備階段(Preparation)

本階段主要任務在於建置一個單純且不受干擾之測試環境，並可操控與掌握各項控制因子，諸如越獄(JB/root)與否、系統平台、系統版本、App介面及行動裝置管理等，可作為檢測時成為考量因素操控使用。

此外，準備階段也應考量攻擊者可能的攻擊角度與手法，透過App內外部、網路或裝置存取以及可能的檢測方法與工具等。最後，在進行檢測前也應提供檢測者關於App之開發事項或心得等，有助於檢測者更有效率的完成檢測，並且有系統地進行檢測流程步驟以及完善檢測紀錄與報告。

4.5.2.2. 情報蒐集(Intelligence Gathering)

本階段主要任務在於盡可能彙整蒐集App相關完整資訊，並進行2項分析分別為「環境分析(Environment Analysis)」與「架構分析(Architectural Analysis)」，並著重在架構與技術部分，尚不需要考慮其他App特定需求。在環境分析中主要進行商業案例(Business Case)與相關利害關係者(Stakeholder)的辨識與分析，此外也要分析內部的流程與架構。

在結構分析中則針對App的網路介面、使用資料、相關資源的溝通、會談管理(Session Management)及越獄(JB/root)偵測等，以及運行環境(Runtime Environment)的行動裝置管理(MDM)、越獄(JB/root)與作業系統版本進行分析，最後也針對分析後端服務(Backend Service)的App伺服器(Application Server)、資料庫(Database)與防火牆(Firewall)等進行資訊蒐集與分析。

4.5.2.3. 威脅建模(Threat Modeling)

本階段主要的任務在於辨識出App的各項威脅，也就是以攻擊者的角度辨識可能影響App的威脅並進行評估，透過前一個情報蒐集的階段分析與確

認環境與架構後，辨識評估出風險並進行高低排序以發展改善對應的控制措施。

在威脅的參考基準上可以從工業局公告之「行動應用 App 基本資安基準」、「OWASP Mobile Top 10」或其他組織持續更新公告之安全威脅資訊，以了解基本與當前必須要注意的威脅。在此威脅建模階段應產出辨識出的威脅紀錄文件、威脅造成風險的高低排序與對應的控制措施。以下為本威脅建模階段的任務步驟：

●切割與群集 App。

●辨識各元件之威脅。

●比較對應所有辨識出的威脅。

●評鑑所有的風險。

●發展辨識出風險的對應控制措施。

●制定測試計畫案例。

4.5.2.4. 弱點分析(Vulnerability Analysis)

本階段主要任務在於根據先前階段的測試計畫案例，以辨識 App 的各項弱點。在執行測試計畫案例時可以從 3 類技術方法進行處理，分別為「靜態方法」、「動態方法」及「鑑識方法」。在靜態方法中主要採用「逆向工程分析」與「自動化與人工原始碼分析」。

4.5.2.5. 弱點評估(Vulnerability Assessment)

透過使用自動化的弱點檢測工具，進行不同目的之檢測方法，用於發現並識別漏洞與風險。在偵測漏洞與風險後，進行評估與統計報告，最後提供開發人員進行 App 改善之建議參考。

4.5.3. App 檢測方法

本指引在檢測方法論主要以 Cigital-TouchPoint Model 為基礎，透過「需求制定與使用者案例」、「架構設計」、「檢測計畫」、「程式碼」、「測試與結果」及「檢測回饋」等面向，其可分為 7 類最佳檢測實務，分別為「源碼檢測(工具基礎)」、「風險分析」、「滲透測試」、「以風險基礎之安全測試」、「濫用案例」、「安全需求制定」與「安全操作」等(詳見圖 35 所示)。本指引參考 App 檢測方法基礎發展之 Cigital-TouchPoing Model 方法論，並以此模型之構面作為本指引之工具採集、編撰及說明之歸納基礎框架，有助於開發人員在不同用途與需求，作為其 App 安全檢測之應用。

資料來源：https://www.cigital.com/presentations/ARA10.pdf(查詢時間：2016/10/1)

圖35 Cigital – TouchPoint Model 方法論

前述提出多個可以工具驗證確認安全之面向，在蒐集、編撰及說明相關檢測工具後，並彙整對應在不同階段流程中，且說明及建議使用不同之檢測方法。本指引以此為基礎提供開發人員在不同情況下可選擇合適之方法進行檢測。

- 在 App 弱點檢測分析階段，可採用靜態方法、動態方法及鑑識方法
  - 靜態方法主要在於透過程式碼分析驗證與確認與安全規格、標準規範等對應差異點與改善點所在，可透過如 dex2jar、otool、androwarn 及 Flawfinder 等工具進行分析。
  - 動態方法則在於網路行為監視分析、App 行為監視與分析、檔案存取行為監視與分析或特徵功能呼叫之監視與分析等，可透過封包擷取工具、Intents 或觀察檔案前後變化獲得動態分析結果。
  - 鑑識方法則在於透過時間軸以及 Log 分析，驗證與確認 App 之安全規格與行為合理性。
- 在分析方法部分，可區分為「動態分析」與「靜態分析」兩個面向
  - 靜態分析主要是在不執行 App 的情況下，在不同開發階段以自動化方法或是人工方式檢測分析確認是否滿足需求規格，並產出錯誤報告以進行修正。

    在靜態分析常用的方法可為檢視(Inspection)、結構化逐步審查(Structured Walkthrough)與主動審查(Active Review)，而其分析標的可為 App 產品計畫、需求規格文件與程式碼等內容。
  - 動態測試則是在 App 完成後進行分析檢測，並需規劃測試案例(Test Case)設計、執行方式及結果蒐集評估。在動態測試中一般使用黑箱測試(Black Box)與白箱測試(White Box)兩種情境作為檢測案例，並透過此一測試檢查 App 的各項介面、功能與非功能需求及相關執行邏輯等正確性。

4.5.3.1. 白箱測試

白箱測試又稱結構測試(Structural Testing)或邏輯驅動測試(Logic-Driven Testing)，是一般常稱的「Code Review」也就是透過 App 結

構與邏輯進行檢測案例的設計及執行。在白箱測試中針對待測的控制流(Control-Flow)與資料流(Data-Flow)提出檢測案例，藉此設計不同的檢測路徑以完成白箱測試。

4.5.3.2. 黑箱測試

黑箱測試又稱為功能測試(Functional Test)、資料驅動測試(Data-Driven Test)或輸入/輸出驅動測試(Input/Output Driven Test)，目前常被稱作「滲透測試(Penetration Test)」，指的是在不了解檢測標的的內部邏輯與結構下執行檢測的工作，形同把 App 放在黑盒子中透過控制輸入資料方式，檢測其行為與輸出結果。黑箱測試不同於白箱測試已知結構與邏輯，因此只能透過規格(Specification)來設計檢測案例。

本指引設計一份簡要的主測試檢核表，作為開發或測試者在針對每個檢測項目可進行快速的條列檢核，並搭配各項目對應之報告或文件進行確認或改善。當檢核未通過時，應要求該項目負責人進行修正改善，並提出改善報告以修正檢核結果，相關 App 自主測試檢核表(範例)詳見表 17 所示。

表17 App 測試檢核表(範例)

檢核與否	檢測編號	檢測項目	檢核工具	檢核結果
	4.1.1.1.1	行動應用 App 應於可信任來源之行動應用 App 商店發布	文件檢核	■通過 □未通過
	4.1.1.1.2	行動應用 App 發布說明	文件檢核	■通過 □未通過
	4.1.1.2.1	行動應用 App 應於可信任來源之行動 App 商店發布更新	文件檢核	■通過 □未通過
	4.1.1.2.2	行動應用 App 應提供更新機制	文件檢核	■通過 □未通過
	4.1.1.2.3	行動應用 App 應於安全性更新時主動公告	文件檢核	■通過 □未通過
	4.1.1.3.1	行動應用 App 問題回報	文件檢核	■通過

檢核與否	檢測編號	檢測項目	檢核工具	檢核結果
				□未通過
	4.1.2.1.1	行動應用 App 敏感性資料蒐集聲明	文件檢核	■通過 □未通過
	4.1.2.1.2	行動應用 App 提供使用者拒絕敏感性資料蒐集機制	文件檢核	■通過 □未通過
	4.1.2.2.1	行動應用 App 應於使用敏感性資料前，取得使用者同意	文件檢核	■通過 □未通過
	4.1.2.2.2	行動應用 App 應提供使用者拒絕使用敏感性資料之權利	文件檢核	■通過 □未通過
	4.1.2.2.3	行動應用 App 如採用通行碼認證，應主動提醒使用者設定較複雜之通行碼	文件檢核	□通過 ■未通過
	4.1.2.2.4	行動應用 App 應提醒使用者定期更改通行碼	文件檢核	□通過 ■未通過
	4.1.2.3.1(1)	檢查行動應用 App 是否於可信任之 App 商店或行動應用 App 內聲明。	文件檢核	■通過 □未通過
	4.1.2.3.1.(2)	檢查行動應用 App 是否於可信任之 App 商店或行動應用 App 內取得使用者同意。	文件檢核	■通過 □未通過
	4.1.2.3.1.(3)	檢查在未聲明或未取得使用者同意敏感性資料儲存的情況下，行動應用 App 是否未儲存敏感性資料。	文件檢核	■通過 □未通過
	4.1.2.3.2(1)	檢查行動應用 App 是否提供使用者拒絕儲存敏感性資料之選項。	文件檢核	■通過 □未通過
	4.1.3.3.2(2)	檢查在使用者拒絕敏感性資料儲存的情況下，行動應用 App 是否未儲存敏感性資料於行動裝置。	文件檢核	■通過 □未通過
	4.1.2.3.3	行動應用 App 儲存之敏感性資料，應僅用於其使用聲明之用途	文件檢核	■通過 □未通過
	4.1.2.3.4(1)	檢查行動應用 App 是否未檢出將敏感性資料儲存於網頁暫存檔或自定	文件檢核	■通過 □未通過

檢核與否	檢測編號	檢測項目	檢核工具	檢核結果
		義暫存檔。		
	4.1.2.3.4.(2)	檢查行動應用 App 是否未檢出將敏感性資料儲存於系統日誌或自定義日誌。	文件檢核	■通過 □未通過
	4.1.2.3.5(1)	檢查行動應用 App 是否採用金鑰有效長度為 128 位元、192 位元或 256 位元之先進加密標準(AES)。	文件檢核	■通過 □未通過
	4.1.2.3.5.(2)	檢查行動應用 App 是否採用金鑰有效長度為 112 位元或 168 位元之三重資料加密演算法(Triple DES)。	文件檢核	■通過 □未通過
	4.1.2.3.5.(3)	檢查行動應用 App 是否採用 SHA-256。	文件檢核	■通過 □未通過
	4.1.2.3.6	檢查行動應用 App 是否儲存敏感性資料於其他行動應用 App 預設無法存取之區域。	文件檢核	■通過 □未通過
	4.1.2.3.7	檢查行動應用 App 之程式碼或其他封裝之檔案內容，是否未檢出密碼、身分驗證資訊或對稱式加解密演算法之金鑰。	文件檢核	■通過 □未通過
	4.1.2.4.1(1)	檢查行動應用 App 是否採用 TLS 1.1(含)以上版本加密協定傳輸敏感性資料。	Santoku	□通過 □未通過
	4.1.2.4.1.(2)	檢查行動應用 App 是否採用金鑰有效長度為 2048 位元(含)以上之 RSA 加密演算法，或採用金鑰有效長度為 224 位元(含)以上之橢圓曲線加密演算法(Elliptic Curve Cryptography)。	Santoku	■通過 □未通過
	4.1.2.4.1.(3)	檢查行動應用 App 是否採用金鑰有效長度為 128 位元、192 位元或 256 位元之進階加密標準(AES)，或採用金鑰有效長度為 112 位元或 168 位	文件檢核	■通過 □未通過

檢核與否	檢測編號	檢測項目	檢核工具	檢核結果
		元之三重資料加密演算法(Triple DES)。		
	4.1.2.5.1.(1)	檢查行動裝置內之不同行動應用App間，分享敏感性資料前，是否於行動應用App內或可信任之App商店聲明。	文件檢核	■通過 □未通過
	4.1.2.5.1.(2)	檢查行動裝置內之不同行動應用App間，分享敏感性資料前，是否於行動應用App內或可信任之App商店取得使用者同意。	文件檢核	■通過 □未通過
	4.1.2.5.1.(3)	檢查在未聲明或未取得使用者同意敏感性資料分享的情況下，行動應用App是否未分享敏感性資料予行動裝置內之不同行動應用App。	文件檢核	■通過 □未通過
	4.1.2.5.2.(1)	檢查行動應用App是否提供使用者拒絕分享敏感性資料之選項。	文件檢核	■通過 □未通過
	4.1.2.5.2.(2)	檢查在使用者拒絕敏感性資料分享的情況下，行動應用App是否未分享敏感性資料。	文件檢核	■通過 □未通過
	4.1.2.5.3	檢查分享敏感性資料之行動應用App，是否限定特定行動應用App可存取敏感性資料。	文件檢核	■通過 □未通過
	4.1.2.6.1	行動應用App如涉及儲存使用者敏感性資料，應提供使用者刪除之功能	文件檢核	■通過 □未通過
	4.1.3.1.1	檢查行動應用App內於付費前，是否主動通知使用者，且資訊至少包含付費資源名稱、數量、金額及付費方式。	文件檢核	□通過 ■未通過
	4.1.3.1.2.(1)	檢查行動應用App內於付費時，是否提供使用者拒絕付費之選項。	文件檢核	□通過 ■未通過
	4.1.3.1.2.(2)	檢查在使用者拒絕付費的情況下，行動應用App是否未進行付費。	文件檢核	□通過 ■未通過

檢核與否	檢測編號	檢測項目	檢核工具	檢核結果
	4.1.3.2.1	檢查行動應用 App 於付費時,是否提供身分認證機制。	文件檢核	□通過 ■未通過
	4.1.3.2.2	檢查行動應用 App 於付費後,是否提供查詢交易紀錄之管道,且交易紀錄至少包含付費資源名稱、付費時間及付費金額之紀錄。	文件檢核	□通過 ■未通過
	4.1.4.1.1	如行動應用 App 存取與個人資料相關之敏感性資料,檢查行動應用 App 是否提供認證機制。	Santoku Snoop-it	■通過 □未通過
	4.1.4.1.2	如行動應用 App 存取與個人資料相關之敏感性資料,檢查行動應用 App 是否提供身分授權機制。	Santoku Snoop-it	■通過 □未通過
	4.1.4.2.1.(1)	檢查行動應用 App 是否採用有效長度為 128 位元(含)以上之交談識別碼。	文件檢核	■通過 □未通過
	4.1.4.2.1.(2)	檢查行動應用 App 使用之交談識別碼是否未與時間、使用者上傳資料、具規則性之數字或字串有直接關聯或難以偽造。	文件檢核	■通過 □未通過
	4.1.4.2.1.(3)	檢查行動應用 App 使用之交談識別碼是否具備逾時失效機制。	文件檢核	■通過 □未通過
	4.1.4.2.2.(1)	檢查行動應用 App 是否使用伺服器憑證仍於有效期間內、未被註銷(Revoke),且憑證之主體名稱與主體別名包含連線之伺服器網域名稱。	Santoku	■通過 □未通過
	4.1.4.2.2.(2)	檢查行動應用 App 是否使用憑證綁定(Certificate Pinning)方式驗證並確保連線之伺服器為行動應用 App 開發人員所指定。	Santoku	■通過 □未通過
	4.1.4.2.3	如行動應用 App 使用安全加密傳輸協定,檢查行動應用 App 是否驗證並	文件檢核	■通過 □未通過

檢核與否	檢測編號	檢測項目	檢核工具	檢核結果
		確保伺服器憑證為行動作業系統內建可信任之憑證機構、政府機關、企業簽發。		
	4.1.4.2.4.(1)	如不符合檢測編號 4.1.4.2.2 或 4.1.4.2.3 之技術要求，檢查行動應用 App 是否未與伺服器進行連線與傳輸敏感性資料。	文件檢核	■通過 □未通過
	4.1.4.2.4.(2)	檢查行動應用 App 是否使用符合檢測編號 4.1.4.2.2 與 4.1.4.2.3 之憑證與伺服器進行連線與傳輸敏感性資料。	文件檢核	■通過 □未通過
	4.1.5.1.1(1)	檢查行動應用 App 是否未針對其他行動應用 App 或行動作業系統之檔案，在未授權情況下，嘗試進行查詢、新增、修改、刪除、存取遠端服務、提權等行為。	Santoku MobSF AppUSE	■通過 □未通過
	4.1.5.1.1(2)	檢查行動應用 App 是否未包括可導致其他行動應用 App 或行動作業系統，發生未預期錯誤、資源明顯耗損、重新啟動或關閉等行為。	Santoku MobSF AppUSE	□通過 ■未通過
	4.1.5.1.2	行動應用 App 資訊安全漏洞	Santoku MobSF AppUSE	■通過 □未通過
	4.1.5.2.1	行動應用 App 應使用適當且有效之完整性驗證機制，以確保其完整性。	文件檢核	■通過 □未通過
	4.1.5.3.1	檢查行動應用 App 引用之函式庫是否不存在已知安全性漏洞。	文件檢核、 MobSF	□通過 ■未通過
	4.1.5.4.1.(1)	檢查行動應用 App 是否針對使用者輸入字串驗證型別。	文件檢核、 Santoku	■通過 □未通過
	4.1.5.4.1.(2)	檢查行動應用 App 是否針對使用者輸入字串驗證長度。	文件檢核、 Santoku	□通過 □未通過

檢核與否	檢測編號	檢測項目	檢核工具	檢核結果
	4.1.5.4.2.(1)	檢查行動應用 App 是否過濾導致 SQL Injection 之字串。	文件檢核、Santoku	■通過 □未通過
	4.1.5.4.2.(2)	檢查行動應用 App 是否過濾導致 JavaScript Injection 之字串。	文件檢核、Santoku	□通過 □未通過
	4.1.5.4.2.(3)	檢查行動應用 App 是否過濾導致 Command Injection 之字串。	文件檢核、Santoku	■通過 □未通過
	4.1.5.4.2.(4)	檢查行動應用 App 是否過濾導致 Local File Inclusion 之字串。	文件檢核、Santoku	■通過 □未通過
	4.1.5.4.2.(5)	檢查行動應用 App 是否過濾導致 XML Injection 之字串。	文件檢核、Santoku	■通過 □未通過
	4.1.5.4.2.(6)	檢查行動應用 App 是否過濾導致 Format String Injection 之字串。	文件檢核、Santoku	■通過 □未通過
	4.1.5.4.2.(7)	檢查行動應用 App 是否過濾導致 Intent Injection 之字串。	文件檢核、Santoku	■通過 □未通過

資料來源：本計畫整理

為使開發人員執行測試階段需要從事的安全活動有所依循，本指引整理「測試階段安全檢核表」詳見附件 5。

4.6.部署維運階段

本階段目標在於確認行動應用 App 之相關資訊系統之運作安全，在行動應用 App 開發末期將以檢測分析報告為依據，針對行動應用 App 相關系統環境進行部署安全管理，以此強化行動應用 App 之相關系統安全性。

在伺服端部分以雲端形式提供使用者服務時，建議應注意 ISO/IEC 27001、雲端安全聯盟(Cloud Security Alliance, CSA)的 STAR(Security, Trust& Assurance Registry)驗證或歐洲雲端服務聯盟星級驗證(EuroCloud Star Audit, ECSA)相關標準，作為伺服端系統建置或租用之參考。

4.6.1. 安全維運

4.6.1.1. 安全組態管理

行動應用 App 常因雲端服務模式之需求，多設計搭配後端系統提供使用者獲得所需的服務。因此在伺服端系統之相關安全性應注意包含「網路管理」、「系統管理」、「模組元件管理」及「伺服器管理」等，在伺服端系統可採用以滲透測試方式進行檢測，相關可參考的組織來源有：OWASP(Open Web Application Security Project)、IECOM(the Institute for Security and Open Methodologies)及 SANS(System Administration, Networking ,and Security Institute)等開放之相關參考文件，在具體可用的工具可參考有 Bastille(Linux/UNIX 環境)、Microsoft Baseline Security Analyzer(微軟 Windows 環境)及組態基準(Configuration Baseline)，並配合滲透測試工具可作為伺服端安全檢測之全面性考量。

●系統管理

伺服端系統多為 Linux/UNIX 或微軟 Windows 系統，原廠都會提供系統 patch 檔或是 Service Pack 的更新安裝包，並關閉不需使用之系統服務，且不安裝來源不明或具風險之相關第三方軟體。

●網路管理

在網路管理部分應注意整體伺服端主機之網路配置，在防火牆部分應注意關閉不使用的服務與 Port 並確認其運作狀態。

●伺服器管理

伺服器管理泛指網頁伺服器、資料庫系統或 App 後端系統等，並須注意使用者帳號密碼與權限議題、過高的連接權限設置、相關伺服器的更新與設置、採用加密連線設置及相關行為監控稽核等。

●模組元件管理

在採用第三方模組元件時，應注意其更新版本、安全通告及來源信任等議題，避免伺服端系統在建置開發時採用具風險之模組元件，造成後端系統之危害。

●伺服應用程式

在伺服應用程式部分應注意帳號密碼不應以明文方式存在於其源碼或設置檔中，並且應移除進行除錯、維護與測試用之相關後門、帳號與偵錯用源碼等。此外，應注意執行之系統權限應以適當權限，避免不必要之存取。

4.6.1.2. 安全部署管理

根據前述之「系統管理」、「網路管理」、「伺服器管理」、「模組元件管理」及「伺服應用程式管理」等項目類別，進行各個類別的部署管理並具體採用前述之工具與檢核表進行安全項目自我檢測，以完成伺服端之安全部署管理。

4.6.2. 事故管理

在部署維運階段雖已進行各類別項目之檢核與設置，但仍有可能遭遇資訊安全事件的發生，因此在此風險考量下仍應建立相關安全通報程序、人員控管與相關因應責任程序等，最後應從資安事件結果建立回饋學習機制，並蒐集相關資料以利未來在維運管理上之改善參考。列出相關程序與描述，提供開發人員可作為參考。更詳細的資料可參考 ISO/IEC 27001 與 ISO/IEC 20000 等標準。

●資訊安全通報及應變作業。

●資訊安全弱點回報分析措施。

●資訊安全改善與矯正措施。

4.6.3. 弱點管理

在弱點管理的機制上，應該制定整體弱點管理政策、程序落實、執行與後續追蹤及改善等，並進行相關弱點稽核與紀錄。在弱點管理上可建置或安裝相關管理系統，如防毒軟體、防火牆及入侵偵測防禦系統等，協助相關弱點之即時偵測防禦與追蹤紀錄管理。當弱點被偵測且分析後，應進行相關後續修補與更新動作。

為使開發人員執行部署維運階段需要從事的安全活動有所依循，本指引整理「部署維運階段安全檢核表」詳見附件 6。

5. 行動應用 App 資安檢測實務

5.1.App 資安檢測實務前言

本指引建議開發人員發展之行動應用 App 安全檢測活動應由團隊開發人員以外測試人員或第三方單位人員進行檢測，透過靜、動態分析及黑、白箱等測試方式可針對行動應用 App 的安全性議題分別進行檢測，有助於消除開發人員自我開發忽略安全風險之盲點。

此外，獨立測試人員可由測試工程方法蒐集與探討相關惡意或有疑慮之函式庫、線上服務介面或相關第三方程式模組等，並且研擬相關測試方案、表單、報告及改善方法，有助於團隊之分工合作與行動應用 App 提升改版之效率與品質。

本指引以軟體測試工程階段為依據，使用者實務上可在「需求階段」、「設計階段」、「開發實作階段」、「測試階段」及「部署維運階段」進行不同需求與目的之測試，開發過程可採用靜態分析在不直接執行 App 情況下，以人工或自動化方法進行評估，檢核其是否滿足相關需求制定規格，作為改善及錯誤修正之依據。

執行分析方法可包含檢視、結構化逐步審查及主動審查等，並針對建構計畫、需求規格、設計規格、程式碼及測試計畫書等 App 產品相關資料。在本章節主要著重說明在 App 程式撰寫初步完成後，採用動態測試自動化工具，透過黑、白箱方式進行使用案例安全性測試，並顯示分析資料及評估結果。

開發人員在實務上開發人員除了自行執行單元測試工作外，建議整體測試計畫作業可交由其他專責測試人員進行測試，並以較有程式設計經驗開發人員較能判斷解讀與回報相關 App 風險或問題。

實務測試上也應制訂完整測試計畫並配合開發階段時程進行相關測試，在測試過程應隨時檢視測試計畫書以確認或改善測試計畫。以下簡述各階段測試程序之步驟：

●測試規劃：開發專案初期(設計階段以前)應規劃擬定相關測試計畫書，並載明測試目標與範圍及其相關方法。

●測試準備：準備測試工具、環境、正常使用及誤用案例，以順利執行測試計畫。

●App 測試：執行 App 測試並記錄相關過程與回報問題。

●評估控制：監控與檢討測試計畫，並評估其執行成效。

儘管在各階段可視需求進行 App 測試，但在最後產出 App 仍須透過持續整合發展，以完善 App 之安全規格需求。在一個 App 測試計畫執行中，開發人員應在 App 測試與計畫文件間進行多次的測試、記錄、評估、檢討與改善。

在測試流程中，如果發生程式碼有所改變或無法編譯產出時，團隊應共同思考如何解決問題，並在解決問題後記錄作為後續參考使用。本指引提出多種 App 檢測工具，如多合一行動安全檢測框架網站系統、整合性檢測分析平台與相關動態、靜態與網路測試工具等。

建議開發人員可在 App 開發過程各階段產出或預期發行時，透過測試計畫的適當規劃(參考第 4 章 SSDLC 各階段安全活動與檢核表)，並採用 Santoku、Snoop-it 或 MobSF 針對 iOS 或 Android APP 進行測試，並以其產出之報告作為未符合安全項目修正改善之依據，並在完成改善後持續進行多次的掃描測試直到確認所有重大或中、高風險安全漏洞均被修正，透過前述持續整合、持續發布與持續部署流程以產出具安全品質之 App 並發行至商店。

此外，在開發團隊進行相關測試/發布時，可能會遭遇幾個問題而造成效率的降低或錯誤的增加。開發人員可能在 App 軟體架構設計、測試環境的變更及工具效能穩定，考量不一定會與測試/品管/發布人員討論；而測試/發布測試/品管/發布人員可能在測試/發布過程，為降低變更的次數或不了解軟體架構運作，也可能由於對功能的了解不足，難以完整描述發布時應具備的相關資訊。

因此，建議 App 開發團隊在執行專案時，應採用安全開發方法，建立良好的軟體構型或額外可採用強化開發人員與測試/品管/發布人員溝通協作的方法，如使用自動化版佈版或採用 DevOps 協作方式，最後可參考本指引提出之資安檢測基準項目作為開發人員與測試/品管/發布人員之共同品質目標之一，以具體完善整個 App 軟體專案的目標與成果。

5.2.App 檢測工具

本指引經廣泛試用及探討網際網路上相關可針對 App 檢測的工具後，篩選出 3 個主要的工具，分別為 Santoku、Snoop-it 與 MobSF 等工具，並可分別針對 Android、iOS 系統進行黑白箱之測試。

以下針對本指引探討之相關檢測工具依其功能性進行分類，以提供開發人員在檢測過程中有效的選擇適用的檢測工具，後續將針對其中整合式工具進行深入探討與測試。本節中將列舉 Santoku、Snoop-it 及 MobSF 等 3 種主要的檢測工具進行簡單介紹。

此外，當開發人員在與測試/品管/發布人員在溝通協作上遭遇困難時，如考慮納入 DevOps 的開發概念，並可額外採用如 HockeyApp 這類的工具。HockeyApp 可供 iOS、OS X、Android 或 Windows 裝置應用程式，以及以 Xamarin、Cordova 及 Unity 跨平台應用程式進行測試，透過 SDK 的整合發布供作測試的版本，藉此將測試版本發予測試人員，並可蒐集未正

常終止的錯誤報告(Crash Report)，了解 App 有瑕疵錯誤的類別或方法，並可加以改善。

在蒐集用戶回饋(Feedback)部分，可以與客戶溝通並針對需求進行討論。在發布(Distribution)部分則可發出 Beta 版本，透過整合的 SDK 可自動檢測更新，提供更新的服務。最後可透過分析指標了解測試人員在不同的測試工具所執行的狀況與內容。

●多合一行動安全檢測框架網站系統

此類網站系統為整合性分析系統，多為提供智慧後端分析並可自動化產生相關報表與結果，多可同時支援 ipa 與 apk 檔案格式。

– Mobile Security Framework(MobSF)。

●整合性檢測分析平台整合性檢測分析平台為透過提供多個檢測工具整合的檢測環境系統，針對不同的檢測目的，如網路封包行為檢測、行動裝置檔案瀏覽、逆向工程工具及惡意程式分析等。

– AppUse：設計用作 Android 的滲透測試平台，並具自有 Android 模擬系統，可在運作時測試觀察使用。

– Santoku：包含一系列開源的安全測試工具，可作為手機資料取證、惡意軟體分析與相關安全測試等。

– Mobisec：行動安全裝置的測試環境，其包含多樣開源安全工具。

●逆向工程與靜態分析工具

– APKTool：用作逆向分析 Android apk 的工具，並可重新編譯打包。

– AndroidBugs：Android 漏洞掃描器，可幫助開發或測試人員掃描潛在的安全漏洞。

– APKInspector：可用作分析 Android 應用程式的圖形化工具。

– JD-GUI：作為逆向與分析 Java 程式碼的工具。

– iRET：iOS 應用程式逆向工具箱，具有多款工具可操作使用。

– class-dump：可將系統應用程式資料讀出並作為分析。

●動態與執行階段分析工具

– Logcat-color：動態查看運作日誌，作為分析使用。

– AndBug：Android 平台的 Dalvik 虛擬機的監視測試工具。

– Drozer：在設備與應用程式搜尋安全漏洞，並進行評估。

– Frida：可注入正在運行的應用程式行程中，進行觀察與監視。

– Snoop-it：在 iOS 進行應用程式安全與動態分析的工具。

– Gdb：分析 iOS 應用程式運作的工具。

– Idb：簡易版的 iOS 滲透測試工具。

●網路分析與伺服端測試

– Wireshark：網路封包截取與分析工具。

– Burp Suite：網路封包截取與分析工具，具備 Proxy 與密碼破解等功能。

– Tcpdump：截取 TCP 協定之封包工具。

– Clarles Proxy：在 MAC 系統下常見封包截取工具，可設置為 Proxy 作為截取封包之用。

– OWASP iMAS：支援 iOS 應用程式安全漏洞掃描之工具。

●安全函式庫檢測

– PublicKey Pinning：可用作檢測憑證相關函式庫的工具。

– Proguard：可瀏覽列出應用程式之函式庫並進行檢查。

– SQL Cipher：作為確認 SQLite 資料庫是否被設置為加密使用。

●行動裝置檔案瀏覽

– iFunbox：支援瀏覽 iOS 系列相關裝置之檔案。

– iTools：同時支援 iOS 與 Android 系統的檔案瀏覽。

5.2.1. Santoku

Santoku 是一個以 Ubuntu Linux 為基礎的整合的工具環境檢測系統。原作者整合多樣化的工具可提供使用者針對手機或 App 進行檢測或分析。Santoku 主要的功能分為 3 個部分，分別為：

●行動裝置 App 鑑識。

●行動裝置 App 惡意軟體分析。

●行動應用安全檢測

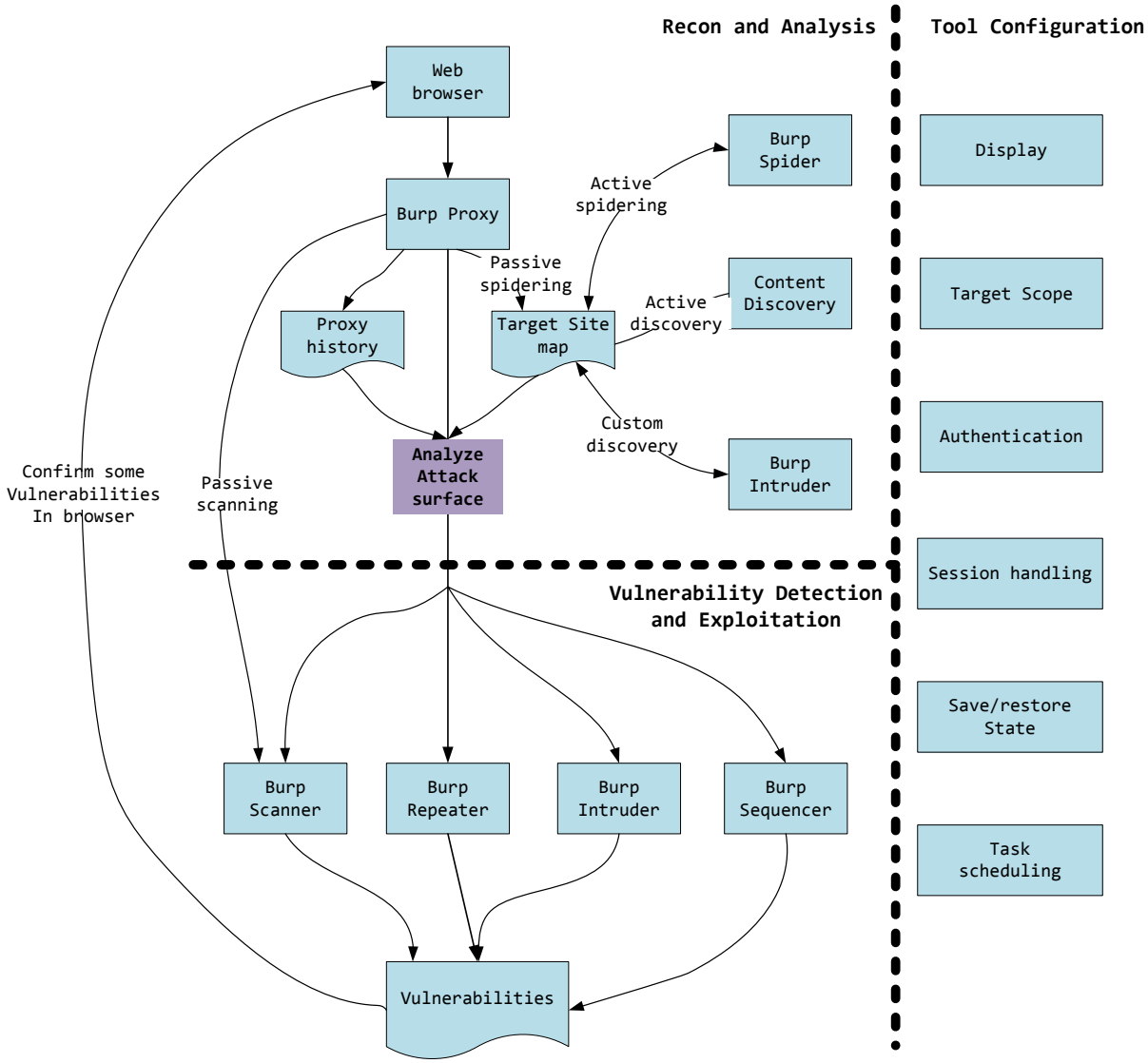
– 適用平台：iOS、Android。

– 工具版本：0.5 版。

資料來源：Santoku

圖36 Santoku 操作環境

本指引主要採用其中 BurpSuite 進行行動應用 App 黑箱之網路檢測，探討其在網路資料傳輸或交換之安全項目。



資料來源：https://portswigger.net/burp/help/suite_usingburp.html（查詢時間：2016/10/1）

圖37 BurpSuite 檢測流程

本指引透過 BurpSuite 可針對以下項目進行檢測：

- 基於客戶端的身分驗證之缺陷。

●客戶端授權違規。

●WebView 的安全設定強化不足(XSS)。

●注入(SQLite 的注入，XML 注入)。

●不安全的傳輸層協議。

●曝光設備的特定標識在攻擊者可見元素。

●Webserver 安全配置錯誤。

●在 API 輸入驗證。

●通過 API 響應做信息暴露。

●繞過業務邏輯漏洞。

●在後端會話失效。

●會話超時保護。

●Cookie 轉置。

●token 創建。

5.2.2. Snoop-it

●Snoop-it 藉由 debugging 和 runtime tracing 功能檢測現有的行動應用 App，以幫助做為動態分析和黑箱安全評估的工具。Snoop-it 允許即時操作任意 iOS 系統架構的行動應用 App，透過一個易於使用的界面，可以繞過客戶端限制或解鎖額外付費的功能和應用程序。藉此測試行動應用 App 在安全驗證或資料安全之相關項目。Snoop-it 主要的功能如下所列：

●Monitoring。

●Analysis/Manipulation。

●Simple installation and configuration。

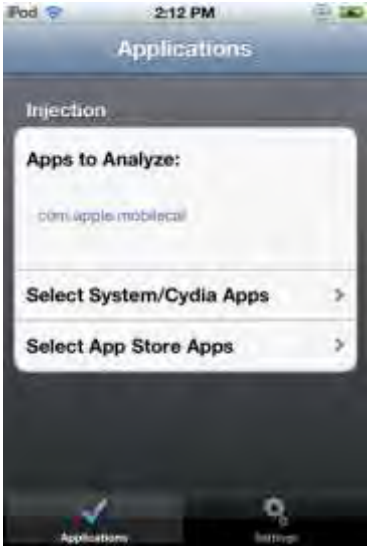
●Easy to use graphical user interface。

●Plenty of filter and search options。

●Detailed description of the XML-RPC web service interface

– 適用平台：iOS。

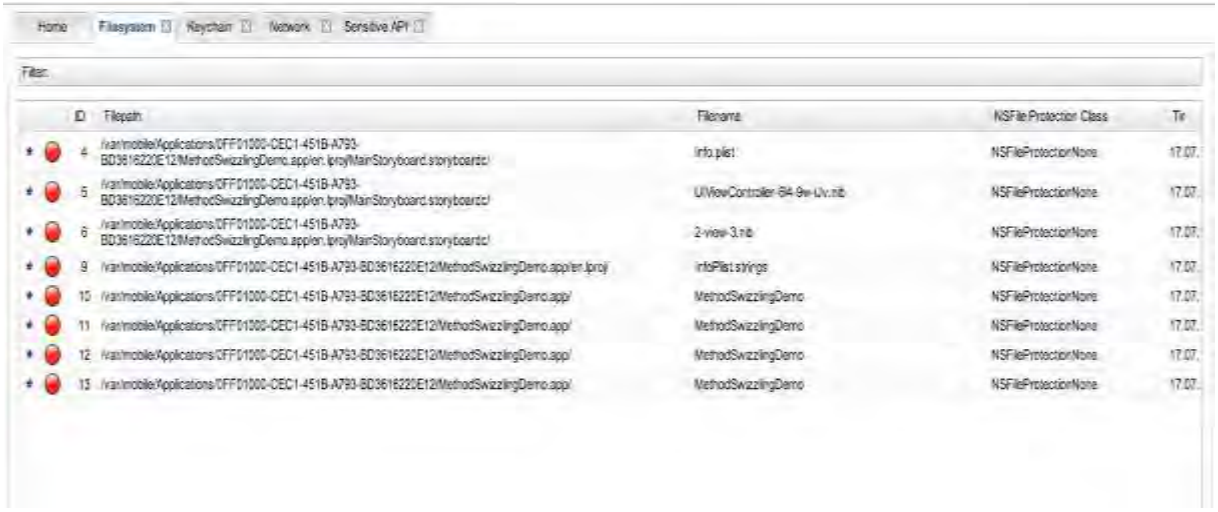
– 工具版本：1.0。



資料來源：Snoop-it

圖38 Snoop-it 操作畫面

Snoop-it 允許查看哪些文件和目錄目前正被應用存取。為了達到這個目的，可以透過 Monitoring 中的 Filesystem 功能進行操作使用，當 App 正在寫入資料庫資料的時候，這個功能可以讓找出資料庫文件的名字，並下載到本地端機器上進階分析。



資料來源：Snoop-it

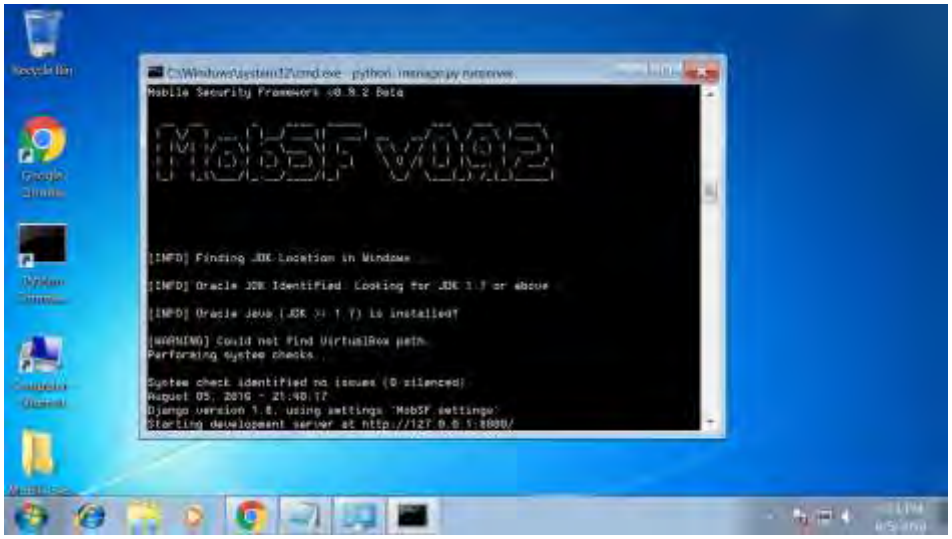
圖39 Snoop-it 針對 App 資料庫進行擷取分析

5.2.3. MobSF

Mobile Security Framework(MobSF)是一個行動 App 分析安全框架，具備多個功能可針對行動應用 App(Android 版/iOS 版)進行分析，這個測試框架能夠執行靜態和動態(僅支援 Android 版本)的分析。

可用於 Android 和 iOS 上並提供有效和快速的安全分析和支援二進制(APK＆IPA)壓縮的原始碼。MobSF 還可以執行 Web API 的安全性測試與 API 模糊測試工具，可以做資訊蒐集，藉以分析安全資料介面，找出如 XXE、SSRF、IDOR，以及有關 session 和 API 限制等邏輯問題以及行動 AppAPI 特定的漏洞。

本指引建議開發人員可採用由 ajinabraham 所發展的 MobSF 工具，有助於多數開發人員可以較快上手進行使用。MobSF 執行畫面詳見圖 40 所示。



資料來源：MobSF

圖40 MobSF 執行畫面

當使用 MobSF 進行行動應用 App 分析時，操作非常簡單(上傳畫面詳見圖 41 所示)。開發人員只要將 APK 或是 IPA 上傳至此 MobSF 分析網站後，即可自動產出相關報告，協助開發人員進行安全檢測之分析。

資料來源：MobSF

圖41 MobSF 上傳行動應用 App 之畫面

本指引透過上傳 Yahoo!天氣 App 進行相關分析，可以產出相關分析報告，詳見圖 42 所示。

資料來源：本計畫整理

圖42 行動應用 App 分析報告畫面

5.3.實務檢測應用

在前述章節已簡介本指引建議採用的檢測工具，分別為 Santoku、Snoop-it 及 MobSF，本指引以 MobSF 自動化工具檢測環境為實務檢測應用，以某電子支付 App 為檢測範例，此 App 提供行動支付、理財管理與相關轉帳、信用卡、收款、繳費及卡券管理等功能服務，本指引透過取得其 apk 與 ipa 檔案，進行實務操作分析。相關的分析步驟與結果如以下範例。

5.3.1. 行動應用 App 安全檢測實務_Android 平台(範例一)

表18 行動應用 App 安全檢測實務_Android 平台(範例一)

【行動 App 資訊】

1. 檔案名稱：testsamlpe01.apk
2. 主要功能：行動支付
3. 取得權限：
   - 聯絡人
     讀取您的聯絡人
   - 電話
     讀取手機狀態和識別碼
     讀取通話紀錄
   - 相機
     拍攝相片和影片
   - 裝置 ID 和通話資訊
     讀取手機狀態和識別碼
   - 其他
     關閉其他 App
     完整網路存取權
     查看網路連線
     安裝捷徑
     解除安裝捷徑

【自我基本檢測目標與流程】

本檢測實務主要係提供開發人員應用於行動 App 開發完成後的初步基本安全檢測，藉以善盡開發人員安全開發的基本責任。

1. 檢測開始前針對待測之行動應用 App 進行基本版本、權限等公開資訊分析。(若本身即為開發人員則已對 App 有相關熟悉度)
2. 透過自動化檢測工具/環境，分別進行靜、動態分析及 Web API 的安全性檢測。
3. 根據其所產生之報告中，發掘問題點的特定項目檢測。針對問題警訊，依據檢測項目不同，其選擇適用之檢測工具及方法。

【檢測環境】:(含名稱、版本等)

1. 工具運行作業系統：Mac OS X EI Capitan v.10.11.6
2. 檢測工具/環境：Mobile Security Framework(MobSF) v0.9.2 Beta(依工具要求環境建置)
3. 行動裝置模擬器：MobSF_VM_0.2.vbox (Samsung Galaxy S4, Android OS v.4.4.2)

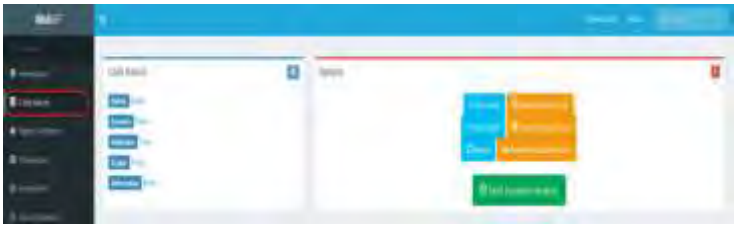
【檢測方法】:

1. ■黑箱測試 □白箱測試	2. ■靜態分析 ■動態分析
【工具官方網址】: ● https://github.com/ajinabraham/Mobile-Security-Framework-MobSF(查詢時間:2016/10/1) ● https://www.python.org/downloads (查詢時間: 2016/10/1) ● http://www.oracle.com/technetwork/java/javase/downloads (查詢時間: 2016/10/1) ● https://www.virtualbox.org/wiki/Downloads (查詢時間: 2016/10/1)	
【相關參考網址】: ● https://github.com/ajinabraham/Mobile-Security-Framework-MobSF/wiki (查詢時間: 2016/10/1)	

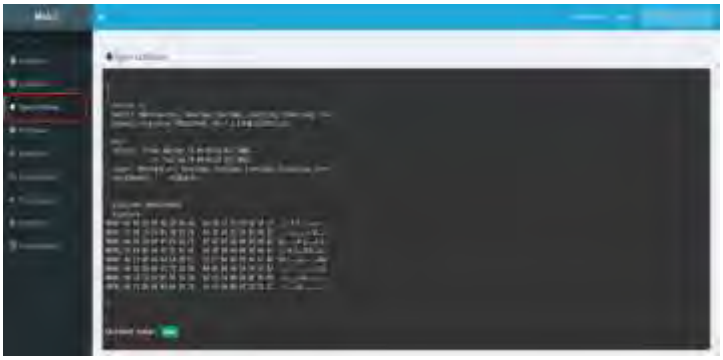
◆ 檢測操作步驟

靜態分析(MobSF)	
 MobSF 開啟畫面	【步驟一】 上傳待測行動應用 App - 將欲檢測的行動 App(App)上傳或拖曳至該測試平台。 - 支援檔案格式為 apk、ipa 檔。

靜態分析- Information(Dashboard)



靜態分析- Code Nature



靜態分析- Signer Certificate

靜態分析- Permissions

【步驟二】

靜態分析-產生測試結果

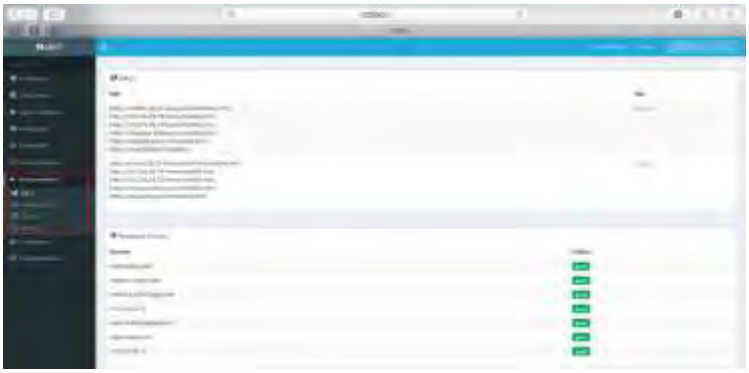
■ App 上傳後，系統即會自動開始進行靜態分析，其檢測項目有：

- ●Information(Dashboard)
- ●Code Nature
- ●Signer Certificate
- ●Permissions
- ●Android API
- ●Security Analysis (Manifest Analysis, Code Analysis, File Analysis)
- ●Reconnaissance (URLs, Malware Check, Emails, Strings)
- ●Componets (Activities, Services, Receivers, Providers, Libraries, Files)

■ 另可下載 Java Code, Smali Code 及 AbdroidMainfest.xml 檔檢視，以進行進階分析。

靜態分析-Android API

靜態分析- Security Analysis



靜態分析- Reconnaissance

靜態分析- Componets

【步驟二】

靜態分析-產生測試結果

■ App 上傳後，系統即會自動開始進行靜態分析，其檢測項目有：

- ●Information(Dashboard)
- ●Code Nature
- ●Signer Certificate
- ●Permissions
- ●Android API
- ●Security Analysis (Manifest Analysis, Code Analysis, File Analysis)
- ●Reconnaissance (URLs, Malware Check, Emails, Strings)
- ●Componets (Activities, Services, Receivers, Providers, Libraries, Files)

■ 另可下載 Java Code, Smali Code 及 AbdroidMainfest.xml 檔檢視，以進行進階分析。

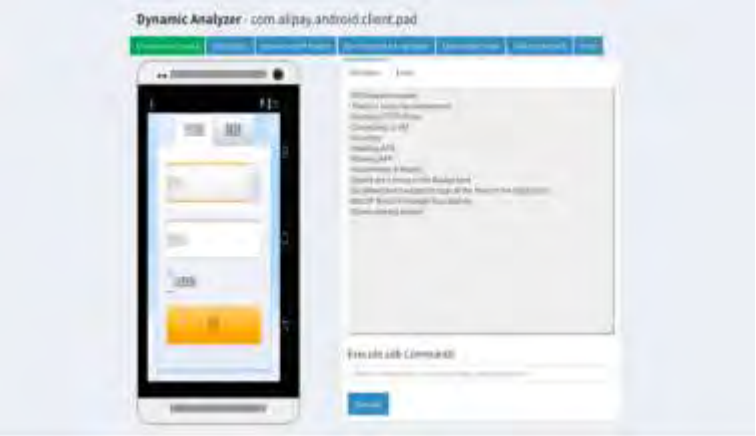
動態分析(MobSF)

動態分析- 開啟畫面

【步驟三】

動態分析-載入測試平台

■ 完成靜態分析後，點擊”Start Dynamic Analysis”進行動態分析(需事先完成 Android 模擬器，及 VM IP, Host/Proxy IP, VM UUID 及 Snapshot UUID 等環境設定)，進入動態分析頁面，並同時載入 Android 模擬器的快照。



動態分析- 測試環境

【步驟四】

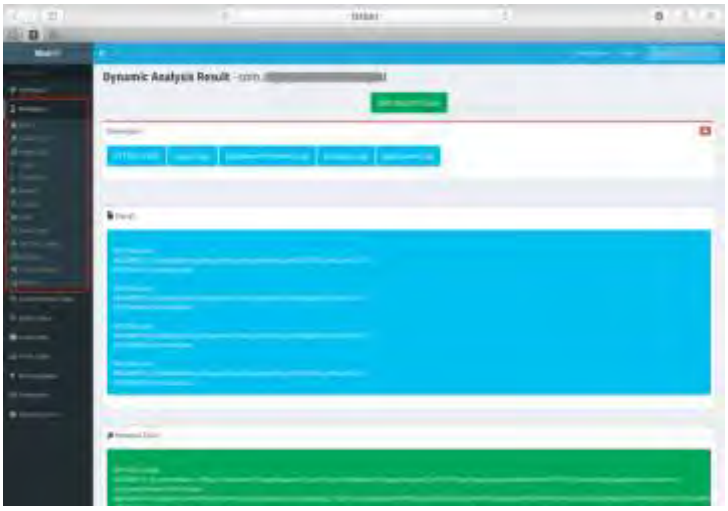
動態分析-測試環境創建

■ 完成載入後，點擊”Create Environment”建立動態測試環境。此時會載入並同時執行行動應用 App。此時可依序進行動態測試項目：

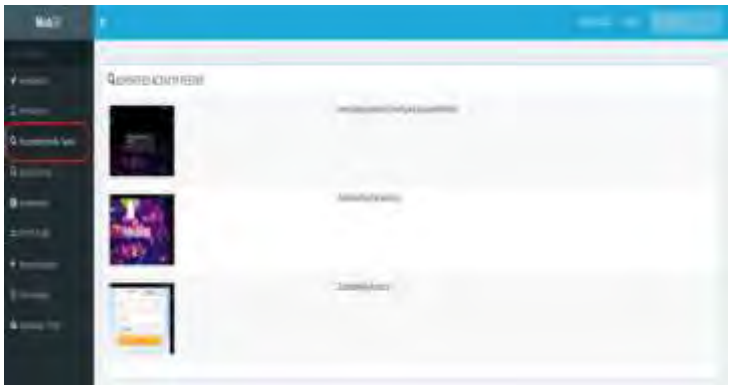
- ●Exported Activity Tester
- ●Activity Tester

■ 另動態分析時，同時可針對 Android 模擬器中行動應用 App 測試情形進行畫面擷取。

動態分析- Information



動態分析- API Monitor



動態分析- Exported Activity Tester

【步驟五】

動態分析-產生測試結果

■ 檢測項目結果有:

- ●Information
- ●API Monitor (File IO, Network Calls, Binder Calls, Crypto, Device Info, Base64, Content, SMS, Device Data, Dex Class, Loader, Reflection, System Manger, Process)
- ●Exported Activity Tester
- ●Activity Tester
- ●Screenshots
- ●HTTPs Traffic
- ●Reconnaissance (URLs, Malware Check, Emails)
- ●File Analysis (SQLite Database, XML Files, Other Files)

■ 並可下載 HTTPs Traffic、Logcat Log、Droidmon API Monitor、Dumpsys Logs 及 Application Data 等原始資料進行進階分析。

動態分析- Activity Tester

動態分析- Screenshots、HTTPs Traffic

動態分析- Reconnaissance

動態分析- File Analysis

【步驟五】

動態分析-產生測試結果

■ 檢測項目結果有:

- ●Information
- ●API Monitor (File IO, Network Calls, Binder Calls, Crypto, Device Info, Base64, Content, SMS, Device Data, Dex Class, Loader, Reflection, System Manger, Process)
- ●Exported Activity Tester
- ●Activity Tester
- ●Screenshots
- ●HTTPs Traffic
- ●Reconnaissance (URLs, Malware Check, Emails)
- ●File Analysis (SQLite Database, XML Files, Other Files)

■ 並可下載 HTTPs Traffic、Logcat Log、Droidmon API Monitor、Dumpsys Logs 及 Application Data 等原始資料進行進階分析。

Web API 分析(MobSF)

Web API 分析- 開啟畫面

【步驟六】

Web API 分析-選取測試項目

■點擊動態分析完成畫面之”Star Web API Fuzzer”執行行動應用 App 之 Web API 應用架構的進階分析，測試之項目有：

- ●Information Gatering
- ●Security Headers
- ●IDOR
- ●Session Handling
- ●SSRF
- ●XXE
- ●Path Travrsal
- ●Rate Limit Check

Web API 分析- 測試結果

【步驟七】

Web API 分析-產生測試結果

■根據測試產生的結果報告，進行檢測項目的解讀與判斷。

特定項目檢測

- 針對特定項目之檢測，由於各檢測項目特性各異，建議可依表 16-App 測試檢核表及表 18-本指引建議工具與「行動應用 App 基本資安檢測基準」項目對應表，尋找適用之檢測工具執行特定項目的進階檢測。

5.3.2. 行動應用 App 安全檢測實務_iOS 平台(範例二)

表19 行動應用 App 安全檢測實務_iOS 平台(範例二)

【行動應用 App 資訊】

1. 檔案名稱：testsamlpe02.ipa
2. 主要功能：行動支付
3. 取得權限：官網未提供。

【自我基本檢測目標與流程】

本檢測實務主要係提供開發人員應用於行動 App 開發完成後的初步基本安全檢測，藉以善盡開發人員安全開發的基本責任。

1. 檢測開始前針對待測之行動應用 App 進行基本版本、權限等公開資訊分析。(若本身即為開發人員則已對 App 有相關熟悉度)
2. 透過自動化檢測工具/環境，分別進行靜及動態分析。
3. 根據其所產生之報告中，發掘問題點的特定項目檢測。針對問題警訊，依據檢測項目不同，其選擇適用之檢測工具及方法。

【檢測環境】：(含名稱、版本等)

1. 工具運行作業系統：
   - Mac OS X EI Capitan v.10.11.6
   - iOS 7.1.2
2. 檢測工具/環境：
   - Mobile Security Framework(MobSF) v0.9.2 Beta(依工具要求環境建置)
   - Snoop-it v.1.0.10(依工具要求環境建置)

【檢測方法】： 3．■黑箱測試 □白箱測試	4．■靜態分析 ■動態分析

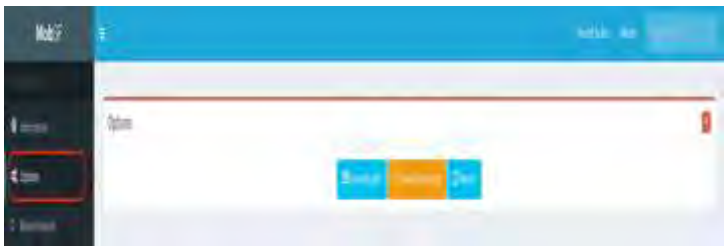
【工具官方網址】：

- https://github.com/ajinabraham/Mobile-Security-Framework-MobSF/(查詢時間: 2016/10/1)
- https://www.python.org/downloads/(查詢時間: 2016/10/1)
- http://www.oracle.com/technetwork/java/javase/downloads/(查詢時間: 2016/10/1)
- https://www.virtualbox.org/wiki/Downloads/ (查詢時間: 2016/10/1)
- http://osxdaily.com/2014/02/12/install-command-line-tools-mac-os-x/ (查詢時間: 2016/10/1)
- http://repo.nesolabs.de/ (Cydia) (查詢時間: 2016/10/1)

【相關參考網址】：

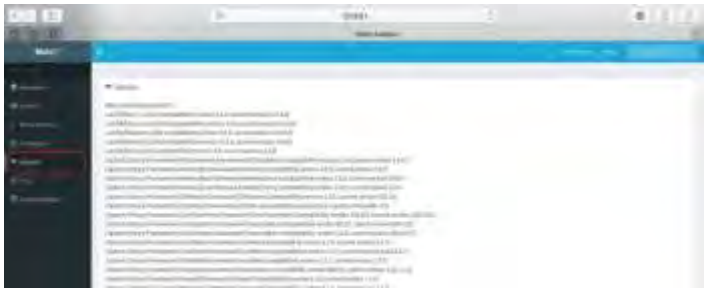
- https://github.com/ajinabraham/Mobile-Security-Framework-MobSF/wiki (查詢時間: 2016/10/1)
- https://itunes.Apple.com/cn/App/id333206289?mt=8 (查詢時間: 2016/10/1)
- http://resources.infosecinstitute.com/iOS-Application-security-part-9-analyzing-security-of-iOS-Applications-using-snoop-it/ (查詢時間: 2016/10/1)

◆ 檢測操作步驟

靜態分析(MobSF)	
 MobSF-開啟畫面	【步驟一】 上傳待測行動應用 App 將欲檢測的行動 App(App)上傳或拖曳至該測試平台。 支援檔案格式為 apk、ipa 檔。
  靜態分析- Information 靜態分析- Option	【步驟二】 靜態分析-產生測試結果 ■ App 上傳後，系統即會自動開始進行靜態分析，其檢測項目有： ●Information ●Option ●Binary Analysis ●File Analysis ●Libraries ●Files ■ 另可下載 Info.plist 及 Class Dump，以進行進階分析。

靜態分析- Binary Analysis

靜態分析- File Analysis



靜態分析- Libraries

靜態分析- Files

【步驟二】

靜態分析-產生測試結果

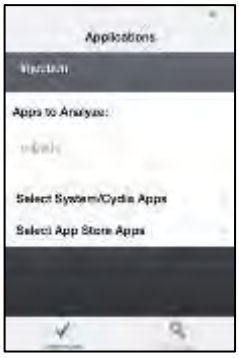
■ App 上傳後，系統即會自動開始進行靜態分析，其檢測項目有：

- ●Information
- ●Option
- ●Binary Analysis
- ●File Analysis
- ●Libraries
- ●Files

■ 另可下載 Info.plist 及 Class Dump，以進行進階分析。

動態分析(Snoop-It)

動態分析-開啟畫面(瀏覽器)



動態分析-開啟畫面(行動裝置)

【步驟三】

動態分析-載入測試平台

■ 基於 MobSF 檢測平台僅支援 Android 作業系統的動態分析，故範例中另提供另一套專門針對 iOS 平台的檢測工具 Snoop-it。

- ●行動裝置：先開啟 Snoop-it，進行相關環境設定，記下其所提供之 IP 位址及端口，同時載入待測的行動應用 App。
- ●電腦端：開啟網頁瀏覽器，引導至 Snoop-it 所提供之 IP 位址及端口。

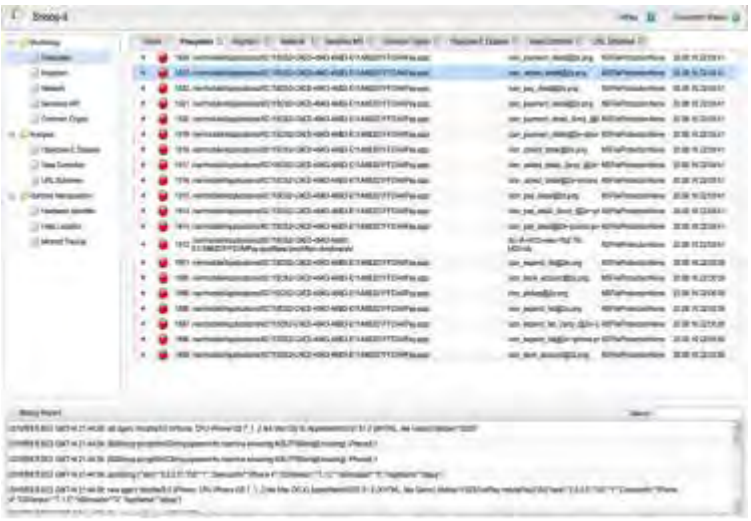
動態分析- 測試環境

動態分析- 測試環境

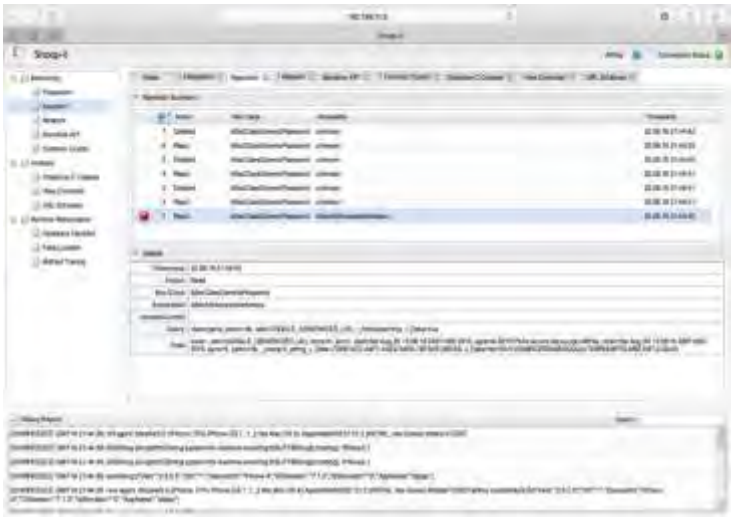
【步驟四】

動態分析-進行即時檢測

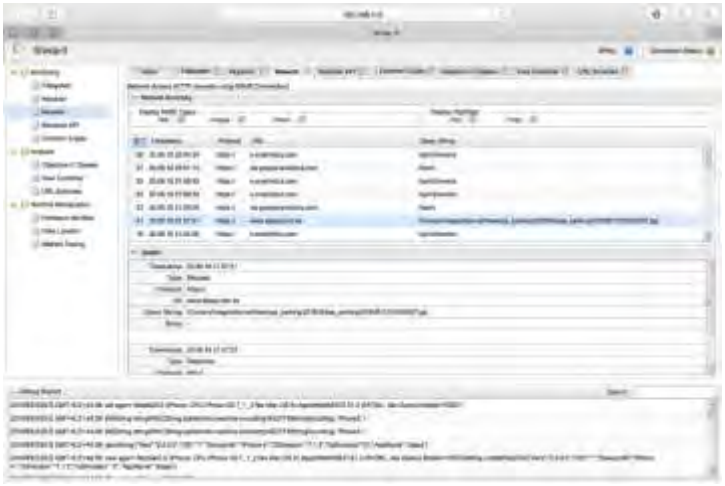
- 行動裝置：此時切換至待測之行動應用 App 的畫面。
- 電腦端：重新整理網頁瀏覽器，此時即載入待測之行動應用 App。



動態分析- Monitoring(Filesystem)



動態分析- Monitoring(Keychain)



動態分析- Monitoring(Network)

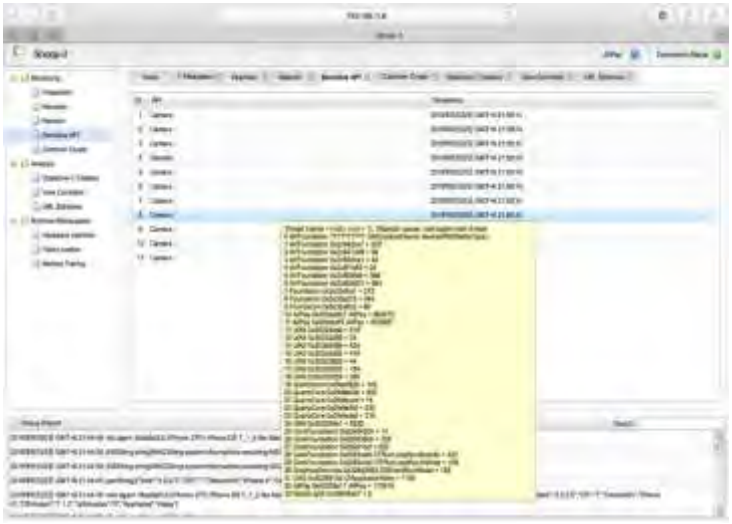
【步驟四】

動態分析-進行即時檢測

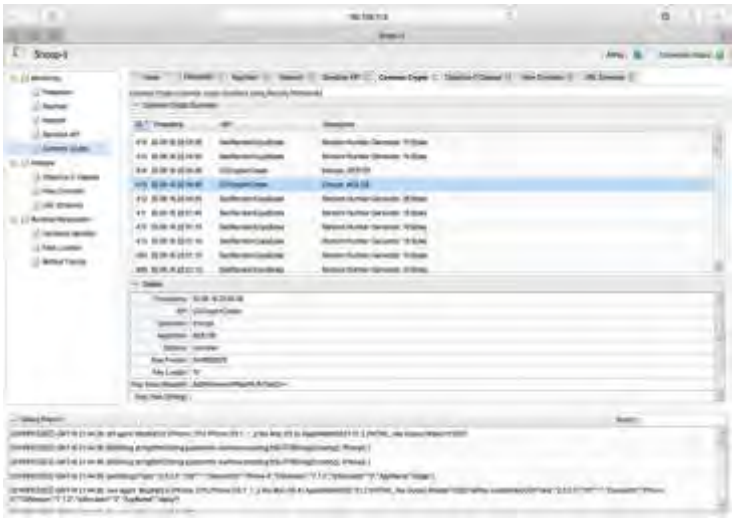
■ 檢測功能區分成 Monitoring 及 Analysis 等 2 大類有:

- ●Monitoring
  - — Filesystem
  - — Keychain
  - — Network
  - — Sensitive API
  - — Common Crypto
- ●Analysis
  - — Object-C Classes
  - — View Controller
  - — URL Schemes

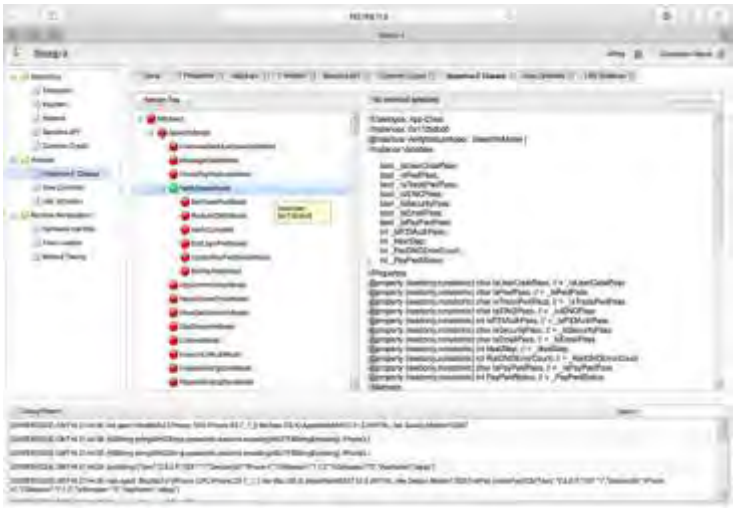
■ 並可運用其他工具下載 Keychain 等資料進行進階分析。



動態分析- Monitoring(Sensitive API)



動態分析- Monitoring(Common Crypto)



動態分析- Monitoring(Network)

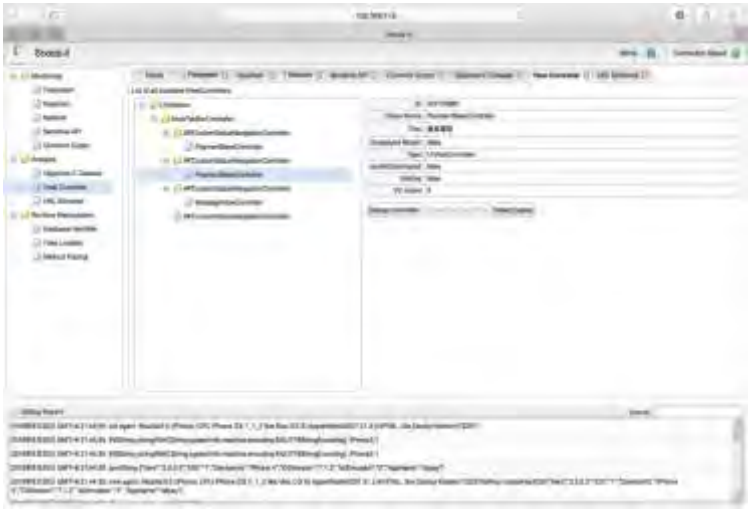
【步驟四】

動態分析-進行即時檢測

■ 檢測功能區分成 Monitoring 及 Analysis 等 2 大類有:

- ●Monitoring
  - — Filesystem
  - — Keychain
  - — Network
  - — Sensitive API
  - — Common Crypto
- ●Analysis
  - — Object-C Classes
  - — View Controller
  - — URL Schemes

■ 並可運用其他工具下載 Keychain 等資料進行進階分析。



動態分析- Analysis(View Controller)

動態分析- Analysis(URL Schemes)

【步驟四】

動態分析-進行即時檢測

■ 檢測功能區分成 Monitoring 及 Analysis 等 2 大類有:

- ●Monitoring
  - — Filesystem
  - — Keychain
  - — Network
  - — Sensitive API
  - — Common Crypto
- ●Analysis
  - — Object-C Classes
  - — View Controller
  - — URL Schemes

■ 並可運用其他工具下載 Keychain 等資料進行進階分析。

特定項目檢測

針對特定項目之檢測，由於各檢測項目特性各異，建議可依表 16 -App 測試檢核表及表 18-本指引建議工具與「行動應用 App 基本資安檢測基準」項目對應表，尋找適用之檢測工具執行特定項目的進階檢測。

5.4.「行動應用 App 基本資安檢測基準」各構面與開發最佳實務工具對應

本指引以工業局公告「行動應用 App 基本資安檢測基準」與相關安全規範等之相關安全項目，對應本指引提出可應用之檢測工具，彙整對應表詳見表 20 所示，提供開發人員引用檢測參考。

表20 本指引建議工具與行動應用 App 基本資安檢測基準」項目對應表

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
4.1.1. 行動 App 發布安全	4.1.1.1. 行動 App 發布	基本資安檢測基準	4.1.1.1.1 行動應用 App 應於可信任來源之行動 App 商店發布	文件檢核
		基本資安檢測基準	4.1.1.1.2 行動應用 App 應於發布時說明欲存取之敏感性資料、行動裝置資源及宣告之權限用途(CPA-01)	文件檢核
		共通安全開發實務準則	CPA-01：於行動應用程式商店提供及應用程式內實作提供隱私權政策說明連結，說明欲存取之敏感性資料、行動裝置資源及宣告權限用途。	文件檢核
		共通安全開發實務準則	CPA-02：行動應用 App 實際權限與於行動平台商店提供及應用程式宣告終端使用者授權約定(EULAs)、應用程式說明、程式內部通知及與 CPA-01 於欲存取之敏感性資料、行動智慧裝置資源及宣告權限用途一致。	文件檢核
		共通安全開發實務準則	CPA-03：應於行動應用 App 上架前確保內部軟體品質流程及版本控制均已實作完成。	文件檢核
		共通安全開發實務準則	CPA-04：確保應用程式規格遵循行動應用商店，如蘋果的 App Store 和 Google Play 規範的規格。	文件檢核
		共通安全開發實務準則	CPA-05：執行應用程式在產品上架期間中的版本控制。	文件檢核
	4.1.1.2. 行動 App 更新	基本資安檢測基準	4.1.1.2.1 行動 App 應於可信任來源之行動 App 商店發布更新	文件檢核
		基本資安檢測基準	4.1.1.2.2 行動 App 應提供更新機制	文件檢核
		基本資安檢測基準	4.1.1.2.3 行動應用 App 應於安全性更新時主動公告	文件檢核

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
		共通安全開發實務準則	CPB-01：行動應用 App 應提供功能性與安全性更新，以因應新功能加入、發現漏洞及平台安全性提升之需求。	文件檢核
		共通安全開發實務準則	CPB-02：應建立內部行動應用 App 變更管理程序。	文件檢核
		共通安全開發實務準則	CPB-03：所有修改版本，應檢查是否已符合行動應用平台商店，如 App Store 和 Google Play 建立的更新的要求。	文件檢核
	4.1.1.3. 行動 App 安全性問題回報	基本資安檢測基準	4.1.1.3.1 行動應用 App 開發人員應提供回報安全性問題之管道	文件檢核
		基本資安檢測基準	4.1.1.3.2. 行動應用 App 開發人員應於適當期間內回覆問題並改善	文件檢核
		共通安全開發實務	CPC-01：於可信任之應用程式商店或行動應用程式內，提供聯絡網頁、電子郵件、電話或其他類型聯絡方式。	文件檢核
		共通安全開發實務	CPC-02：行動應用 App 開發團隊，應關注作業系統平台或安全社群或常見弱點通報(CVE)之相關弱點通報，改善行動應用 App 安全性。	文件檢核
4.1.2. 敏感性資料保護	4.1.2.1. 敏感性資料蒐集	基本資安檢測基準	4.1.2.1.1 行動應用 App 應於蒐集敏感性資料前，取得使用者同意	文件檢核
		基本資安檢測基準	4.1.2.1.2 行動應用 App 提供使用者拒絕敏感性資料蒐集機制	文件檢核
		共通安全開發實務	CPD-01：需求階段需要識別計畫開發行動應用 App 蒐集、處理及利用敏應資料類別(例如：密碼，個人資料、地理位置、財務資訊及錯誤日誌等)及行動應用 App 平台與發行國的隱私法令要求，定義安全性需求。	文件檢核
		共通安全開發實務	CPD-02：建立預計提供給使用者的敏感資料蒐集範圍、類別、保存期限、使用地點及生命週期處理政策。	文件檢核

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
		共通安全開發實務	CPD-03：在設計階段依行動應用 App 需求識別需蒐集、處理、儲存、利用、傳送及刪除敏感性資料類別與內、外部媒體儲存，以使用者同意及最低揭露原則設計權限安全性控制措施，如身分驗證、API 呼叫、加密儲存及傳送等安全性。	文件檢核
		共通安全開發實務	CPD-04：檢查在同一資料堆疊(Stack)多個蒐集敏感資料不同蒐集來源(如原生 AP+ WebKit 的 HTML)是否重疊並衝突，如有需解決衝突。	文件檢核
		共通安全開發實務	CPD-05：蒐集敏感性資料應實作使用者同意或拒絕選項，提示使用者選擇時機可於(1)安裝時(2)當敏感資料被儲存或傳送前(3)預設設定為關閉同意，需使用者自行開啟。	文件檢核
		共通安全開發實務	CPD-06：應依 CPD-03 實作行動應用 App 存取敏感性資料權限，不要授與過度蒐集不必要敏感性資料權限或於未告知使用者取得同意前於行動應用 App 運作背景蒐集敏感性資料。	文件檢核
	4.1.2.2. 敏感性資料利用	基本資安檢測基準	4.1.2.2.1 行動 App 應於使用敏感性資料前，取得使用者同意	文件檢核
		基本資安檢測基準	4.1.2.2.2 行動 App 應提供使用者拒絕使用敏感性資料之權利	文件檢核
		基本資安檢測基準	4.1.2.2.3 行動 App 如採用通行碼認證，應主動提醒使用者設定較複雜之通行碼	文件檢核
		基本資安檢測基準	4.1.2.2.4 行動 App 應提醒使用者定期更改通行碼	文件檢核
		共通安全開發實務	CPE-01：不要使用設備或與其他 App 共享持久性敏感資料，如設備 ID，作為識別碼，最好採用隨機產生識別碼(參考 CPM-02)，並採用相同的資料最小化權限原則應用在行動應用 App 的 Session ID 及 HTTP Session ID/cookies 等。	文件檢核

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
		共通安全開發實務	CPE-02：追蹤個人使用者應以使用者帳號密碼作為識別使用者，避免使用電話號碼作為識別碼，並實作非基於簡訊驗證的雙因素驗證。	文件檢核
		共通安全開發實務	CPE-03：不得實作未經使用者同意或變更權限，使行動應用 App 可擅自修改使用者資料的行為，包括在使用者無確認情況下刪除或修改使用者連絡人資料、通話記錄、簡訊資料和多媒體簡訊資料的行為。	文件檢核
		共通安全開發實務	CPE-04：謹慎處理行動應用 App 利用敏感性資料如 NFC 實體位置、WIFI 位址、GPS 位置及稽核日誌等，不被非授權存取或傳送到未知遠端目的地。	文件檢核
		共通安全開發實務	CPE-05：敏感性資料於頁面顯示時應依隱碼標準要求處理，如必要顯示明碼需先經使用身分認證及授權。	文件檢核
		共通安全開發實務	CPE-06：加密傳送敏感性字串查詢，並使用有 XSRF token(token)保護的 POST 指令發送，以防止重送攻擊(replay attack)。	Santoku
		共通安全開發實務	CPE-07：行動應用 App 本地及伺服器端可使用代碼化，以取代完整金融帳號或其他個人資料，如銀行帳戶及信用卡號。	文件檢核
	4.1.2.3. 敏感性資料儲存	基本資安檢測基準	4.1.2.3.1 行動 App 應於儲存敏感性資料前，取得使用者同意	文件檢核
		基本資安檢測基準	4.1.2.3.2 行動 App 應提供使用者拒絕儲存敏感性資料之權利	文件檢核
		基本資安檢測基準	4.1.2.3.3 行動 App 儲存之敏感性資料，應僅用於其使用聲明之用途	文件檢核
		基本資安檢測基準	4.1.2.3.4 行動 App 應避免將敏感性資料儲存於暫存檔或紀錄檔中	文件檢核
		基本資安檢測基準	4.1.2.3.5 敏感性資料應採用適當且有效之金鑰長度與加密演算法，進行加密處理再儲存	文件檢核

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
		基本資安檢測基準	4.1.2.3.6 敏感性資料應儲存於受作業系統保護之區域，以防止其他 App 未經授權之存取	文件檢核
		基本資安檢測基準	4.1.2.3.7 敏感性資料應避免出現於行動 App 之程式碼	文件檢核
		共通安全開發實務	CPF-01：行動應用 App 如需要儲存敏感性資料需依 CPD-02 規劃於可信任之應用程式商店或行動應用程式內聲明。	文件檢核
		共通安全開發實務	CPF-02：行動應用 App 儲存敏感性資料應實作提示使用者同意及拒絕選項，提示使用者選擇時機可於(1)安裝時(2)當敏感資料被儲存或傳送前(3)預設設定為關閉同意，需使用者自行開啟。	文件檢核
		共通安全開發實務	CPF-03：行動應用 App 應依 CPD-03 規劃實作適當資料儲存區域劃分，如記憶體、暫存檔、資料庫、日誌、全域可讀(公用)、不同敏感資料類別及內、外部儲存媒體，並依最小權限原則設定存取權限及安全機制，如加密、分割或代碼化。	文件檢核
		共通安全開發實務	CPF-04：將敏感性資料儲存於伺服器端取代儲存於行動智慧裝置本地端，前提是安全連線機制已建立及伺服器端儲存的安全機制足夠或優於行動智慧裝置端。	文件檢核
		共通安全開發實務	CPF-05：優先使用行動裝置作業系統提供的 API 實作儲存加密機制，加密模組應通過 FIPS 140-2 認證，使用加密/簽章演算法與金鑰長度與如下： (1)加密金鑰有效長度為 128 位元、192 位元或 256 位元之先進加密標準(AES)演算法。 (2)加密金鑰有效長度為 112 位元或 168 位元之三重資料加密演算法(Triple DES)演算法。 (3)簽章雜湊函數/金鑰長度 SHA-256	文件檢核

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
		共通安全開發實務	CPF-06：實作從加密儲存區域解密資料，需經使用者或系統管理員身分認證及授權機制。	文件檢核
		共通安全開發實務	CPF-07：敏感性資料或包含敏感性資料日誌檔，除非已加密，應避免儲存於與其他行動應用 App 共用或全域可讀寫儲存區域或外部儲存媒體。	文件檢核
		共通安全開發實務	CPF-08：需注意記憶體暫存的敏感性資料，如固定金鑰與密碼的安全性，當不需要時或於一定合理期間應強制清除或使用於一定期間就失效的可變量金鑰替代。	文件檢核
		共通安全開發實務	CPF-09：以設定或程式編寫關閉行動應用 App 本地端及網頁伺服器端暫存檔 HTTPS 流量資料，如日誌/除錯文件、Cookie、明文密碼、密碼 hash 值、網頁歷史、網頁暫存檔及屬性列表等。	文件檢核
		共通安全開發實務	CPF-10：行動應用 App 正式版本應關閉或移除除錯日誌，或避免儲存敏感性資料於未加密日誌儲存區。	文件檢核
		共通安全開發實務	CPF-11：行動應用 App 正式版本避免產生或傳送明文意外中止日誌。	文件檢核
		共通安全開發實務	CPF-12：避免於行動應用 App 之程式碼、二進位碼或其他封裝程式中，直接嵌入密碼、身分驗證資訊或對稱式加解密演算法之密鑰(Keys)、初始化向量(Initial Vector, IV)。	文件檢核
		共通安全開發實務	CPF-13：當提供使用者選擇儲存使用者 ID，來加速日後身分驗證便利性，應實作加密方法，以保護真實 ID，而顯示時，可使用選項有遮罩(Mask)或、代碼化(Token)及雜湊(Hash)等。	文件檢核
		共通安全開發實務	CPF-14：使用設定或編寫程式停用對敏感資料鍵盤輸入資料自動更正(auto-correct)功能。	文件檢核

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
		共通安全開發實務	CPF-15：使用設定或編寫程式停用敏感資料區域的複製及貼上功能。	文件檢核
		共通安全開發實務	CPF-16：當行動應用 App 使用行動智慧裝置攝影鏡頭拍攝敏感性資料，如身分證、財務文件，應僅暫存於記憶體不儲存於本機檔案系統，完成傳輸後即由記憶體清除。	文件檢核
		共通安全開發實務	CPF-11：行動應用 App 正式版本避免產生或傳送明文意外中止日誌。	文件檢核
		iOS 安全開發實務	iOS-01：謹慎使用 Keychain 儲存密碼 (Use the Keychain carefully)	文件檢核
		共通安全開發實務	CPF-17：行動應用 App 有提供標準的設定參數檔函數，但由於駭客取得安裝檔之後，非常容易就可以修改這些標準的參數檔，於是為了加強安全層級，應用程式的設定參數，建議放在安全的地方，例如：編譯至程式碼中，或者進行加密。	文件檢核
		共通安全開發實務	CPF-18：使用行動應用 App Cookies 的安全設定。	文件檢核
		OWASP Mobile 2016 Top 10	M2： Hard-coded credentials on sourcecode 將驗證的機密性資料直接寫入	Santokustring, jdgui, IDA, Hopper, MobSF, AppUSE
		OWASP Mobile 2016 Top 10	M6： Cryptographic Based Storage Strength 加密演算基礎的儲存強度	jdgui, YSO, Qark, AndroBugs
		OWASP Mobile 2016 Top 10	M6： Poor key management process 脆弱金鑰管理流程	jdgui, YSO, Qark, AndroBugs

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
		OWASP Mobile 2016 Top 10	M6： Use of custom encryption protocols 自定義加密協定的使用	jdgui, YSO, Qark, AndroBugs
		OWASP Mobile 2016 Top 10	M2： Unrestricted Backup file 未被限制的備份文件	Santoku,apktool ,Androidmanifest.xml
		OWASP Mobile 2016 Top 10	M2： Unencrypted Database files 未加密資料庫檔案	Santoku iFunBox,adb, idb
		OWASP Mobile 2016 Top 10	M2： Insecure Shared Storage 不安全的共享儲存	Santoku iFunBox,adb, idb
		OWASP Mobile 2016 Top 10	M2： Insecure Application Data Storage 不安全的 App 資料儲存	Santoku adb, idb, iFunbox,BinaryCookieReader
		OWASP Mobile 2016 Top 10	M4： Information Disclosure through Logcat/Apple System Log (ASL) 通過 logcat 的訊息披露/蘋果系統日誌(ASL)	Santoku,CatLog, idb, Snoop-it
		OWASP Mobile 2016 Top 10	M4： Application Backgrounding (Screenshot) App 後台(截圖)	adb, iFunbox
		OWASP Mobile 2016 Top 10	M4： URL Caching (HTTP Request and Response) on cache.db cache.db 上的 URL 暫存(HTTP 請求和響應)	adb, iFunbox
		OWASP Mobile	M4： Keyboard Press Caching 鍵盤按壓暫存	adb, iFunbox

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
		2016 Top 10		
		OWASP Mobile 2016 Top 10	M4： Copy/Paste Buffer Caching 複製/貼上緩衝區暫存	adb, iFunbox
	4.1.2.4. 敏感性資料傳輸	基本資安檢測基準	4.1.2.4.1 行動 App 透過網路傳輸敏感性資料，應使用適當且有效之金鑰長度與加密演算法進行安全加密	Santoku
		共通安全開發實務	CPG-01：行動應用 App 傳輸敏感性資料應規劃傳輸全程全時使用 TLS 1.1 以上加密，維護敏感性資料機密性及完整性。	Santoku
		共通安全開發實務	CPG-02：行動應用 App 應使用作業系統平台或可信任來源提供最新版，TLS API 及安全實作規範，實作全時全程以上安全傳輸管道。	Santoku
		共通安全開發實務	CPG-03：行動應用 App 實作 CPG-02 TLS 加密應使用或於加密模組驗證計畫(CRYPTOGRAPHIC MODULE VALIDATION PROGRAM ,CMVP)適用性清單中加密模組或知名加密演算法並採用足夠長度加密金鑰。 (a)採用金鑰有效長度為 2048 位元(含)以上之 RSA 加密演算法，或採用金鑰有效長度為 224 位元(含)以上之橢圓曲線加密演算法(Elliptic Curve Cryptography)。 (b)採用金鑰有效長度為 128 位元、192 位元或 256 位元之進階加密標準(AES)，或採用金鑰有效長度為 112 位元或 168 位元之三重資料加密演算法(Triple DES)。	文件檢核
		OWASP Mobile 2016 Top 10	M3： Insecure Transport Layer Protocols 不安全的傳輸層協議	Santoku,Burp suite

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
		OWASP Mobile 2016 Top 10	M3： SSL/TLS Weak Encryption SSL/TLS 弱加密	Santoku,testssl.sh, Qualys SSL Labs
		OWASP Mobile 2016 Top 10	M3： Disable certificate validation 禁用憑證驗證	Santokujdgui, YSO, Qark, AndroBugs
		OWASP Mobile 2016 Top 10	M3： Self-signed certificate 自簽名證書	Santoku,jdgui, YSO, Qark, AndroBugs
		OWASP Mobile 2016 Top 10	M4： Exposing Device Specific Identifiers in Attacker Visible Elements 暴露裝置特定辨識符號在攻擊者可視元素中	Santoku,Burpsuite
	4.1.2.5. 敏感性資料分享	基本資安檢測基準	4.1.2.5.1 行動裝置內之不同行動 App 間，應於分享敏感性資料前，取得使用者同意	文件檢核
		基本資安檢測基準	4.1.2.5.2 行動 App 應提供使用者拒絕分享敏感性資料之權利	文件檢核
		基本資安檢測基準	4.1.2.5.3 行動 App 分享敏感性資料時，應避免未授權之行動 App 存取	文件檢核
		共通安全開發實務	CPH-01：將敏感性資料分享給不同行動應用 App 應實作使用者同意或拒絕選項，提示使用者選擇時機可於(1)安裝時(2)當敏感資料被儲存或傳送前(3)預設設定為關閉同意，需使用者自行開啟。	文件檢核
		共通安全開發實務	CPH-02：當使用者已於行動應用 App 選取拒絕分享敏感性資料給其他行動應用 App 選項，不應實作於運作中背景將敏感性資料，以任何形式連線機制傳送給其他應用程式或不明目的地。	文件檢核

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
		共通安全開發實務	CPH-03：在分享敏感性資料前，要注意驗證接收來自或傳送至的目的地是否為信任可靠。	文件檢核
	4.1.2.6. 敏感性資料刪除	基本資安檢測基準	4.1.2.6.1 行動 App 如涉及儲存使用者敏感性資料，應提供使用者刪除之功能	文件檢核
		共通安全開發實務	CPI-01：行動應用 App 儲存敏感性資料，應預先針對各國個人資料保護法或隱私要求及實務需求，定義資料於行動智慧裝置合理最長保存期限，預設為 App 自使用者行動智慧裝置刪除或反安裝同時，持久性敏感資料屆期可詢問是否需要刪除或是永久性加密需求。	文件檢核
		共通安全開發實務	CPI-02：行動應用 App 本地端及遠端伺服器暫存檔、暫存及儲存於記憶體或內外部儲存媒體敏感性資料，應依 CPI-01 定義敏感性資料最長保留期限設計及實作提供使用者選擇清除及永久性刪除功能。	文件檢核
		共通安全開發實務	CPI-03：於行動智慧裝置 NAND flash 記憶體，刪除敏感性資料除呼叫檔案刪除 API 不保證資料可完全被清除，可以加密提高資料被復原難度。	文件檢核
		共通安全開發實務	CPI-04： 受組織管理行動智慧裝置應充分運用行動作業系統提供資料刪除開關(Data Kill Switch)，刪除遺失或離職員工行動智慧裝置上的敏感資料。	文件檢核
		共通安全開發實務	CPI-05：行動應用程式納入 CPI-04「資料刪除開關」時，應實作需經強認證要求始能啟動此功能，以避免遭受攻擊或濫用危及資料安全。	文件檢核
		共通安全開發實務	CPI-06：當行動應用 App 被反安裝時，應依 CPI-02 設計一併移除敏感性資料，並通知雲端或資料中心端刪除使用者資料或同意保留資料。	文件檢核
4.1.3. 付	4.1.3.1. 付費	基本資安檢測基準	4.1.3.1.1 行動 App 應於使用付費資源前主動通知使用者	文件檢核

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
費資源控管安全	資源使用	基本資安檢測基準	4.1.3.1.2 行動 App 應提供使用者拒絕使用付費資源之權利	文件檢核
		共通安全開發實務	CPJ-01：行動應用 App 內於付費前，是否主動通知使用者，且資訊至少包含付費資源名稱、數量、金額及付費方式，並實作提示使用者同意及拒絕選項。	文件檢核
		共通安全開發實務	CPJ-02：行動應用 App 不應有未向使用者明示且未經使用者同意，擅自呼叫行動智慧裝置通信功能，造成使用者費用損失的行為，包括在使用者無確認情況下撥打電話、發送簡訊、發送多媒體簡訊、開啟行動通信網路 連接並收發資料的行為。	文件檢核
		共通安全開發實務	CPJ-03：行動應用 App 開發人員應實作節約使用行網路流量最佳實務，如快速休眠(3GPP 規範)、暫存檔等，以盡量減少對基地台的訊號負荷。	文件檢核
	4.1.3.2. 付費資源控管	基本資安檢測基準	4.1.3.2.1 行動 App 應於使用付費資源前進行使用者認證	文件檢核
		基本資安檢測基準	4.1.3.2.2 行動 App 應記錄使用之付費資源與時間	文件檢核
		共通安全開發實務	CPK-01：有付費資源之行動應用 App 需要執行 CPJ-01 同意選項，需經身分認證後始能呼叫付費 API。	文件檢核
		共通安全開發實務	CPK-02：行動應用 App 完成付費後，應顯示提示該次付費資訊，至少包含付費資源名稱、付費時間及付費金額。	文件檢核
		共通安全開發實務	CPK-03： 行動應用 App 應提供安全查詢交易紀錄之管道，如在行動應用 App 選單內或於行動應用商店，且交易紀錄至少包含付費資源名稱、付費時間及付費金額之記錄。	文件檢核
		共通安全開發實務	CPK-04：行動應用 App 於執行 CPK-01 時如偵測到異常重複認證，如位置顯著發生變化時，使用者語言的變化等，應增加額外驗證措施。如以電子郵件或簡訊發送交易驗證碼至綁定行動智慧裝置。	文件檢核

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
		共通安全開發實務	CPK-05：行動應用 App 移除時，應提示使用者未來仍持續線上支付訂閱資源，應預設取消或供使用者選擇取消或保留選項。	文件檢核
		共通安全開發實務	CPK-06：行動應用 App 取用 Bar Code 或 QR Code 進行下一步交易動作需與用戶確認。	文件檢核
		OWASP Mobile 2016 Top 10	M5： "Bypassing business logic flaws 繞過業務邏輯漏洞"	Santoku,Burp suite
4.1.4. 身分認證、授權與連線管理安全	4.1.4.1. 使用者身分認證與授權	基本資安檢測基準	4.1.4.1.1 行動 App 應有適當之身分認證機制，確認使用者身分	Santoku Snoop-it
		基本資安檢測基準	4.1.4.1.2 行動 App 應依使用者身分授權	Santoku Snoop-it
		共通安全開發實務	CPL-01：行動應用 App 應設計並實作適當身分認證機制，並依使用者身分授權，以防止敏感資料被非授權人員存取。	Santoku Snoop-it
		共通安全開發實務	CPL-02：如使用固定帳號密碼作為唯一身分認證因素，視資料敏感性設定密碼強度，密碼安全性設計選用下列原則： (a)在人類可記憶情況，視處理資料敏感程度不應少於 8~12 位。若搭配交易驗證碼使用則不應少於 4 位。 (b)建議採英數字混合使用，且宜包含大小寫英文字母或符號。 (c)不應訂為相同的英數字、連續英文字或連號數字，預設密碼不在此限。 (d)密碼與使用者帳號不應相同。 (e)密碼連續錯誤達 5 次，應鎖定或間隔 15 分鐘以上時間解鎖。 (f)變更密碼不得與前一次相同。 (g)使用預設密碼首次登入時，應強制變更預設密碼。 (h)密碼最長有效期限。	文件檢核

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
		共通安全開發實務	CPL-03：行動應用 App 應提供使用者安全變更密碼功能。	文件檢核
		共通安全開發實務	CPL-04：避免在行動應用 App 中直接使用設備 ID(UUID)作為使用者身分追蹤識別作為唯一因素，建議以帳號及密碼作為主要因素，UUID 可作為多因素認證的其他因素。	文件檢核
		共通安全開發實務	CPL-05：行動應用程式在處理敏感性資料時可實作多因素認證，加強身分認安全性，除帳號密碼外作為主要因素外，可考慮使用一次性密碼 token(OTP Token)、安全元件(Secure Element)、雙因素驗證、生物特徵(聲紋、指紋、臉部識別)、知識詢問、約定資訊等	文件檢核
		共通安全開發實務	CPL-06：行動應用 App 存取敏感性資料或是呼叫行動裝置資源，如攝影機、麥克風等有地域或網段的限制，可增加 GPS 座標及網段範圍符合作為認證因子之一。	文件檢核
		共通安全開發實務	CPL-07：行動應用 App 如使用滑動可視密碼(Swipe-based)作為密碼，易遭塗抹的攻擊(smudge-attacks)，在設計上應導入允許重複圖案措施因應。	文件檢核
		共通安全開發實務	CPL-08：行動應用 App 應運用作業系統平台提供內建安全機制，如沙箱應用程式及通過身分認證後，始能允許敏感性資料及行動裝置資源，如電話、簡訊、攝影機、GPS、麥克風等。	文件檢核
		OWASP Mobile 2016 Top 10	M5： Remember Credentials Functionality (Persistent authentication) 記住憑證功能(持續驗證)	adb, iFunbox
		OWASP Mobile 2016 Top 10	M5： Client Side Based Authentication Flaws 基於客戶端的身分驗證之缺陷	Santoku adb, Drozer, Cycript, Snoop-it, Burpsuite

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
		OWASP Mobile 2016 Top 10	Client Side Authorization Breaches 客戶端授權違規	Santoku adb, Drozer, Cycript, Snoop-it, Burpsuite
	4.1.4.2. 連線管理機制	基本資安檢測基準	4.1.4.2.1 行動 App 應避免使用具有規則性之交談識別碼	文件檢核
		基本資安檢測基準	4.1.4.2.2 行動 App 應確認伺服器憑證之有效性	Santoku
		基本資安檢測基準	4.1.4.2.3 行動 App 應確認伺服器憑證為可信任之憑證機構、政府機關或企業簽發	文件檢核
		基本資安檢測基準	4.1.4.2.4 行動 App 應避免與未具有效憑證之伺服器，進行連線與傳輸資料	文件檢核
		共通安全開發實務	CPM-01：行動應用 App 實作 CPG-02 TLS 連線，需使用編碼長度為 128 位元(含)以上之交談識別碼。	Santoku
		共通安全開發實務	CPM-02：行動應用 App 連線使用交談識別碼，應以不可預測規則種子產生亂數。	Santoku
		共通安全開發實務	CPM-03：行動應用 App 連線使用交談識別碼，應實作具備逾時失效(Session time-out)機制。	Santoku
		共通安全開發實務	CPM-04：行動應用 App 使用伺服器憑證需仍於有效期間內、未被註銷(Revoke)，且憑證之主體名稱與主體別名包含連線之伺服器網域名稱。	Santoku
		共通安全開發實務	CPM-05：行動 App 是否使用憑證綁定(Certificate Pinning)方式驗證並確保連線之伺服器為行動 App 開發人員所指定。	文件檢核
		共通安全開發實務	CPM-06：行動應用 App 應使用憑證驗證鏈驗證各段連線跳躍(Hop)TLS 憑證為為行動作業系統內建可信任之憑證機構、政府機關、企業簽發公開憑證(Public CA)。	文件檢核
		共通安全開發實務	CPM-07：行動應用 App 應於完成 CPM-06 與伺服器端憑證鏈驗證後，始能傳輸敏感性資料於本地端及遠端伺服器兩端之間。	文件檢核

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
		共通安全開發實務	CPM-08：行動應用 App 的使用者介面應儘可能讓使用者容易查詢到憑證是否有效。	文件檢核
		OWASP Mobile 2016 Top 10	M9：Session invalidation on Backend 在後端會話失效	Santoku,Burp suite
		OWASP Mobile 2016 Top 10	M9：Session Timeout Protection 會話超時保護	Santoku,Burp suite
		OWASP Mobile 2016 Top 10	Cookie Rotation Cookie 轉置	Santoku,Burp suite
		OWASP Mobile 2016 Top 10	M9：Token Creation token 創建	Santoku,Burp suite
4.1.5. 行動 App 碼安全	4.1.5.1. 防範惡意程式碼與避免資訊安全漏洞	基本資安檢測基準	4.1.5.1.1 行動 App 應避免含有惡意程式碼	Santoku MobSF AppUSE
		基本資安檢測基準	4.1.5.1.2 行動 App 應避免資訊安全漏洞	Santoku MobSF AppUSE
		共通安全開發實務	CPN-01：如有程式碼，使用可檢測行動應用 App 原始碼安全檢測工具或人工進行靜態分析，檢視權限並比對是否與 CPD-03 安全設計及使用者設定權限相符(permission mapping analysis)。	文件檢核
		共通安全開發實務	CPN-02：如無程式碼使用「逆向工程法」輔以「自動化與人工原始碼分析」檢視權限並比對是否符合 CPD-03 安全設計及使用者設定權限相符。	文件檢核

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
		共通安全開發實務	CPN-03：行動應用 App 應考量中央處理單元(CPU)、記憶體(Memory)、輸出入(Input/Output)單元能耗問題，找出個別程序、API 或行動智慧裝置服務呼叫過於損耗功率。	文件檢核
		共通安全開發實務	CPN-04：依正常及錯誤使用案例測試計畫，以動態測試行動應用 App 是否會發生發生未預期錯誤、資源明顯耗損、重新啟動或關閉。	文件檢核
		共通安全開發實務	CPN-05：應檢測行動應用內部是否有非由使用者發起非預期與其他應用程式或第三方服務通訊或不明目的地等行為。	文件檢核、MobSF
		共通安全開發實務	CPN-06：行動應用 App 如有包含直譯器程式碼作為輸入或是呼叫第三方 API，需謹慎處理運行中錯誤，避免遭惡意注入攻擊或版取得存取敏感資料權限。	文件檢核、MobSF
		共通安全開發實務	CPN-07 行動應用 App 應防止網頁框架(html frame)技術和連線劫持	文件檢核、MobSF
		iOS 安全開發實務	iOS-03：啟用自動引用計數(Automatic Reference Counting ,ARC)，避免記憶體物件弱點產生。	文件檢核、MobSF
		iOS 安全開發實務	iOS-04：詳細介紹 App Transport Security(ATS)設定。	文件檢核
		Android 安全開發實務	Android-01：Android 版行動應用 App 應謹慎實作 Intents，避免資訊不慎外洩遭惡意運用。	文件檢核、MobSF
		Android 安全開發實務	Android-02：Android 版行動應用 App 應謹慎檢查活動(Activities)狀態，是否如安全設計預期運作，防止意外外洩敏感性資料。	文件檢核、MobSF
		Android 安全開發實務	Android-03：Android 版行動應用 App 實作廣播(Broadcast intent)應設定權限，避免被惡意行動應用 App 偽造元件。	文件檢核、MobSF

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
		Android 安全開發實務	Android-04：Android 版行動應用 App 實作 PendingIntents，應指定為私有(Private)的廣播接收者/活動，明確在 base intent 指定元件名稱，避免接收外部惡意資料輸入。	文件檢核、MobSF
		Android 安全開發實務	Android-05：Android 版行動應用 App 為避免內部的 Services 不被惡意外部行動應用 App 呼叫，可於 AndroidManifests.xml 設定存取權限保護。	文件檢核、MobSF
		Android 安全開發實務	Android-06：Android 版行動應用 App 的敏感資料在不同行動應用 App 間傳遞時，避免使用廣播式 Intent，易遭惡意 App 讀取。	文件檢核、MobSF
		Android 安全開發實務	Android-07： Android 版行動應用 App 應謹慎實作 Content Providers，避免因權限過高或未驗證資料目的地與來源，遭惡意程式注入式攻擊。	文件檢核、MobSF
		Android 安全開發實務	Android-08： Android 版行動應用 App 應謹慎實作檔案權限，除非有必要為任何應用程式建立可以任意讀取或寫入檔案，避免創造全域可讀寫檔案權限，導致敏感性資料外洩。	文件檢核、MobSF
		OWASP Mobile 2016 Top 10	M7： Local File Inclusion through NSFileManager or Webviews 通過 NSFileManager 或 Webviews 瀏覽本地文件內容	Santoku,iDevice, Drozer
		OWASP Mobile 2016 Top 10	M8： Abusing URL schemes 濫用 URL 技術	iFunbox, Clutch, Strings
	4.1.5.2. 行動 App 完整性	基本資安檢測基準	4.1.5.2.1 行動 App 應使用適當且有效之完整性驗證機制，以確保其完整性。	文件檢核
		共通安全開發實務	CPO-01：行動應用 App 程式碼應使用平台供應商核發企業或個人開發人員憑證簽章。	文件檢核

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
		共通安全開發實務	CPO-02：行動應用 App 的程式碼儘量以原生方法撰寫，呼叫已載入記憶體的函式庫，以使惡意的程式碼分析難度提高。	文件檢核
		共通安全開發實務	CPO-03：行動應用 App 程式碼應運用人工或工具使之增加複雜度，並輔以限制除錯器使用、反追蹤、二進位剝離等措施，使惡意人士使用逆向工程方法分析程式碼難度增加。	文件檢核
		共通安全開發實務	CPO-04：行動應用 App 應程式碼避免使用過於簡單安全設計邏輯，可增加如挑戰/回應、OTP、定時回調(callback)、基於密碼學驗證等增加對攻擊者分析程式碼難度。	文件檢核
		共通安全開發實務	CPO-05：行動應用 App 為保障程式碼及敏感性資料完整性，應實作反篡改技術，如程式碼簽章及資料完整性校驗(Checksum)。	文件檢核
		OWASP Mobile 2016 Top 10	M10 ： Reverse Engineering the Application Code 逆向工程的 App 程式碼	SantokuSantoku apktool, dex2jar, Clutch, Classdump, MobSF, AppUSE
		OWASP Mobile 2016 Top 10	M10： Unauthorized Code Modification 未經授權程式碼之修改	Santoku,apktool, Frida, cycript, snoop-it
		OWASP Mobile 2016 Top 10	M10： Debug the Application behavior through runtime analysis 透過運行時之分析調整 App 的行為	Santoku,adb jdwp, jdb, GDB, LLDB
	4.1.5.3. 函式	基本資安檢測基準	行動 App 於引用之函式庫有更新時，應備妥對應之更新版本，更新方式請參酌基本資安檢測基準 4.1.1.行動 App 發布安全	文件檢核、MobSF

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
	庫引用安全	共通安全開發實務	CPP-01：行動應用 App 使用第三方函式庫前，需先確認其是來自可靠來源、有持續更新並經測試沒有漏洞、後端木馬及不明傳送目的地。	文件檢核、MobSF
		共通安全開發實務	CPP-02：行動應用 App 使用第三方 SDK/API，需先確認其是來自可靠來源、有持續更新並經測試沒有漏洞、後端木馬及不明傳送目的地。	文件檢核、MobSF
		OWASP Mobile 2016 Top 10	M7： Insufficient WebView hardening (XSS) WebView 的強化不足(XSS)	Santoku,jdgui, Burpsuite
		OWASP Mobile 2016 Top 10	M8： Abusing Android Components through IPC intents ("exported" and "intent-filter") 透過 Android 組件濫用 IPC intent 物件(“exported”和“intent filter”)	Santoku,apktool,Androidmanifest.xml
	4.1.5.4. 使用者輸入驗證	基本資安檢測基準	4.1.5.4.1 行動 App 應針對使用者輸入之字串，進行安全檢查	文件檢核、Santoku
		基本資安檢測基準	4.1.5.4.2 行動 App 應提供相關注入攻擊防護機制	文件檢核、Santoku
		共通安全開發實務	CPQ-01：行動應用 App 應實作驗證使用者輸入字串資料型別及長度之正確性，避免惡意輸入導致應用程式毀損、緩衝區溢位、各種注入攻擊發生。	文件檢核、Santoku
		共通安全開發實務	CPQ-02：行動應用 App 需要驗證使用者輸入、伺服器端傳入及其他裝置輸入之資料，防止因 Buffer overflow 造成安全性漏洞。	文件檢核、Santoku
		共通安全開發實務	CPQ-03：行動應用 App 需要驗證使用者輸入及伺服器端傳入資料，防止因 Buffer underflow 造成安全性漏洞。	文件檢核、Santoku

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
		共通安全開發實務	CPQ-04：行動應用 App 需要驗證使用者輸入及伺服器端傳入資料，防止因 Integer overflow and underflow 造成安全性漏洞。	文件檢核、Santoku
		共通安全開發實務	CPQ-05：行動應用 App 應實作過濾使用者輸入及伺服器端傳入資料中易導致 SQL Injection 之字串。	文件檢核、Santoku
		Android 安全開發實務	Android-09： 行動應用 App 應實作過濾使用者輸入及伺服器端傳入資料中易導致 Intent Injection 之字串	文件檢核
		共通安全開發實務	CPQ-06：行動應用 App 應實作過濾使用者輸入及伺服器端傳入資料中易導致 Command injection 之字串。	文件檢核
		共通安全開發實務	CPQ-07：行動應用 App 應避免讓使用者傳入文件傳入檔案系統或框架 API，如為必須允許，實作過濾使用者輸入及伺服器端傳入資料中易導致 Local File Inclusion 之字串。	文件檢核
		共通安全開發實務	CPQ-08：行動應用 App 應實作過濾使用者輸入及伺服器端傳入資料中易導致 XML Injection 之字串。	文件檢核
		共通安全開發實務	CPQ-09：行動應用 App 應實作過濾使用者輸入及伺服器端傳入資料中易導致 Format String Injection 之字串。	文件檢核
		共通安全開發實務	CPQ-10：確認行動應用 App 授權功能使用者介面，可正常如預期運作，不論顯示幕大小、橫向或直向，如允許輸入，需可帶出虛擬鍵盤，其顯示與按鍵需能如預期。	文件檢核
		共通安全開發實務	CPQ-11：行動應用 App 提供使用者輸入值儘量可以參數化(Query parameterization)。	文件檢核
		OWASP Mobile 2016 Top 10	M7： Content Providers： SQL Injection and Local File Inclusion 內容提供商：SQL INJECTION 和本地文件之內容	Santoku,Drozer

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
		OWASP Mobile 2016 Top 10	M7： Injection (SQLite Injection, XML Injection) 注入(SQLite 的注入，XML 注入)	Santoku adb, iFunbox, Burpsuite
		OWASP Mobile 2016 Top 10	M1： Excessive port opened at Firewall 打開過多的防火器端口	Santoku,Nmap
4.2. 伺服端基本資安檢測基準		Android 安全開發實務	Android-10：Android 版行動應用 App 的 Webview 的許多 API 已被發現漏洞，如 addJavscriptInterface ，實依最佳安全實務實作 WebView。	MobSF, Santoku
		OWASP Mobile 2016 Top 10	M1： Default credentials on Application Server 應用伺服器上的預設憑據	Web Browser
		OWASP Mobile 2016 Top 10	M1： Exposure of Webservices through WSDL document 經由 WSDL 文件而造成 Webservices 的暴露	Web Browser
		OWASP Mobile 2016 Top 10	M1： Security Misconfiguration on Webserver Webserver 安全配置錯誤	Santoku,Web Browser, Burpsuite
		OWASP Mobile 2016 Top 10	M1： Input validation on API 在 API 的輸入驗證	Santoku,Burp suite
		OWASP Mobile 2016 Top 10	M1： Information Exposure through API response message 透過 API 回應訊息而資料洩漏	Santoku,Burp suite
		伺服器安全開發實務	Server-01 與行動應用 App 連接之所有後端服務伺服器(包含網頁、資料庫及中介等)	

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
			作業系統應有效強化及進行安全設定配置，並持續進行安全性程式修補。	
		伺服器安全開發實務	Server-02：應依 CPD-01 需求保留日誌在安全且不易遭非授權存取或場篡改儲存空間，作為後續突發事件反映及數位取證原始資料。	
		伺服器安全開發實務	Server-03：實施 code review，檢查在行動智慧裝置和網頁伺服器端和其他外部介面之間傳送的有無任何意外敏感性資料傳輸。	
		伺服器安全開發實務	Server-04：與行動應用 App 連接之所有後端服務(Web 服務/ REST)應定期檢測漏洞，如使用靜態程式碼分析器工具測試和模糊工具檢測和發現的安全漏洞，並及時修復高風險漏洞。	
		伺服器安全開發實務	Server-05：與行動應用 App 連接之所有後端服務(Web 服務/ REST)應定期實施滲透測試，發現安全漏洞，並及時修復高風險漏洞。	
		伺服器安全開發實務	Server-06：網頁伺服器需預防網頁掛馬及點擊劫持攻擊。	
		伺服器安全開發實務	Server-07：網頁伺服器使用 Token 防範跨站請求(Cross-site Request Forgery,CSRF)攻擊。	
		伺服器安全開發實務	Server-08：實作網頁伺服器服務安全一般控管原則。	
		伺服器安全開發實務	Server-09：應評估服務容量及水準，限制個別使用者/IP 流量，以降低被分散式阻斷攻擊(DDoS)風險。	
		伺服器安全開發實務	Server-10：參考 CPG-01~CPG-03 實作 TLS 伺服器端設定。	
		伺服器安全開發實務	Server-11：參考 CPM-01~CPM-08 實作會談管理(Session Management)伺服器端設定。	

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
		伺服器安全開發實務	Server-12：使用網路區隔及網路存取控制措施，保護不應被外部存取的內部資源。	

資料來源:本計畫整理

6. 結語

本指引以經濟部工業局「行動應用 App 基本資安規範」為依據，廣納各國最新行動應用 App 安全開發、檢測實務及經濟部工業局最新公告之「行動應用 App 基本資安檢測基準」，進行分析、歸納本指引第 3 章「行動應用 App 安全開發最佳實務」計 128 條實務。

本指引以經濟部工業局所提出之「行動應用 App 基本資安規範」與「行動應用 App 基本資安檢測基準」兩份規範為基礎，發展 App 檢測實務提供開發與測試人員參考。本指引由「行動應用 App 安全開發基本概論」為基礎，討論目前兩大主流平台 Android 與 iOS 之安全功能、開發環境、風險議題，藉此提供開發者團隊了解 App 開發中之安全機制與需注意之安全風險。

本指引在以 App 基本資安檢測基準上發展安全開發實務，其中介紹各安全需求項目之定義、技術要求、對應方法與應用階段情境等，透過每一個項目的說明，提供開發團隊在各安全品質項目上之自我要求認知。在建立相關安全認知後，本指引發展與描述 App 安全開發生命週期，並在各準備、需求、設計、實作、測試與部署維運階段講述開發過程可應用考量之內容方法，有助於開發團隊在不同階段能夠得到有用的開發指引，並對應前述的安全檢測項目說明，獲得具體有用的方法。當開發團隊可能無法具體解決問題時，本指引也提供一定程度的開發邏輯與相關關鍵字，能夠提供開發團隊相當程度的應用。本指引在最後更具體提出建議可應用於不同情境之工具方法，也取其中易用的整合性工具作為簡要示範，使得開發團隊在了解整個安全開發實務後，能夠在工具的檢測使用上得到印證與幫助。

本指引從國內外的文獻探討、開發方法、實務探討與相關表單工具等探討、整理、歸納及實作，主要目的在於提供國內 App 開發團隊在安全品質上能夠更加完善的指引，透過此一指引的發展，將有助於國內 App 開發團

隊能夠有所依循且有所本，並有足夠程度與國際安全標準接軌而不擔心 App 產品推向國際受到阻礙。

即使是 Google 或 Apple 這樣大型企業的 App 產品都曾有漏洞發布，國內的 App 團隊勢必也時常遭遇這類的安全議題困擾，因此一個良好且長期發展的指引是絕對重要的，也是政府在資訊安全政策上必然的發展路線。在政府的大力支持下，透過指引的發布將有助於國內 App 開發團隊在最低付出心力上，取得良好的 App 安全品質，藉此降低可能在金融、個資、醫療與社交等相關糾紛發生，創造美好的政民合作之 App 安全產品發展平台。

再度呼應前言「安全是被設計出來，不是被駭出來」，持續遵守「使用者意願」、「最小權限」及「密不透風」的 3 大安全原則，需要有一個安全開發流程－安全軟體開發生命週期(SSDLC)，協助開發人員藉由安全需求、安全設計、實作安全、測試安全及上線安全，開發出不會濫取使用者資料、能對抗惡意攻擊的行動應用 App，是本指引的期望。

行動應用平台改版頻繁(Android 7.0 及 iOS 10 均已發布)、安全性漏洞層出不窮，相信「行動應用 App 安全開發指引」V1.0 只是一個起點，除了透過業界、學界審查而完備，未來期望能透過開發人員社群等，回饋實務建議以利後續修訂。

附件

附件 1　安全軟體開發生命週期方法論比較

附件 2　行動應用 App 需求階段檢核表

附件 3　行動應用 App 設計階段檢核表

附件 4　行動應用 App 開發實作階段檢核表

附件 5　行動應用 App 測試階段檢核表測試

附件 6　行動應用 App 部署維運階段檢核表