普遍存在之行動裝置應用程式漏洞

江格* 黃昌平* 高培晟* 王勝德*
*光盾資訊 *台大電機系及台大慶齡工業研究中心資安研究室
kuoc@rayaegis.com chris@rayaegis.com peter.cheng@rayaegis.com sdwang@ntu.edu.tw

摘要

隨著科技及網路傳輸的發展，利用手機與無線網路進行資料交換、傳播，甚至商業資料的傳輸，如網路銀行、電子下單與購物網站已日漸普及，在此同時，敏感資訊傳輸的安全性成了重要的課題，以求兼顧科技帶來的便利性與安全性的平衡。

由於多數 App 應用程式為廠商自行開發或外包，因此邁向資訊安全將是一條充滿挑戰的道路。儘管各類手機 App 的資安問題不勝枚舉，我們將以常見的傳輸加密(SSL)為例，以實例證明政府或銀行等大型組織所發佈之 App 軟體並未正確執行 SSL 加密傳輸協定，將可能造成極嚴重的個資外洩。

因多數應用程式 App 開發者沒有資安背景，造成手機應用程式類似的漏洞為數眾多，我們希望藉此機會提醒資安人員正視此類問題的普遍性及嚴重性，除了防止個資外洩，也保護組織的敏感資訊，我們最後會針對此類問題提供有效建議、稽核方針與解決方案。

關鍵詞：App 應用程式、行動裝置漏洞、SSL 加密傳輸協定

壹、手機應用程式安全性探討

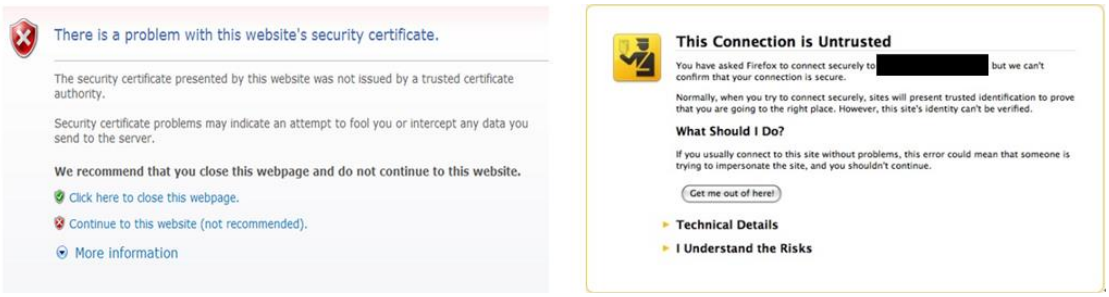
自從 Google 及 Apple 開放應用程式 App 市集以來，市集平台並不對程式執行嚴謹的資安檢查，而是提醒用戶應用程式需要開放的權限，而這種做法衍生出不少資安問題[1]。現在，眾多業者已開發自己的應用程式 App，並開放給用戶下載使用，我們發現其中大多數有嚴重的資安問題。

例如，以 SSL 的實作而言，手機 App 軟體與網頁瀏覽器的安全性等級就有明顯的差異。開發瀏覽器的廠商多為大型公司，如微軟 IE [2]、Mozilla Firefox [3]、Google chrome 等等，相當重視資安議題並投入相當資源，且持續改善發現的問題。相反的，手機 App 軟體多是廠商自行開發，且多為非專業資訊安全人員撰寫，因此大多數並沒有安全檢測的流程或有效預防資安問題[4]。儘管相關資安問題繁多，我們以常見的傳輸加密(SSL)為例，說明目前許多存在於手機 App 的資安漏洞。

無線網路使得攻擊者得以無線電波涵蓋之範圍內，皆可能對使用者傳輸的資料進行截取，此時若傳輸的資料未適當的加密或驗證彼此身分，加上使用安全性較差的公共無線網路，很有可能使用者的資料在網路傳輸過程中被側錄或攔截而造成機敏資料外洩。另外，惡意攻擊者可把各式裝置、手機、電腦，當成代理伺服器，介於用戶與主機之間攔截敏感訊息。SSL 傳輸協議即為保護機敏資料的機密性與完整性，是網頁伺服器和用戶之間加解密方式的安全技術標準，此技術已廣泛用來保護使用者網路傳輸的機敏資

料，所有資料以密文的形式進行傳輸，可防止在資料在傳輸過程中被非法截取或篡改。SSL 憑證則用於建立瀏覽器和網站伺服器之間的安全通道，讓用戶可分辨伺服器是否可以信任、是否存在中間人攻擊。

一般瀏覽器已包含許多重點網站的 SSL 憑證資訊預期清單，以防範使用者遭受中間人攻擊。如果網站伺服器提供的憑證有問題，使用者(client side)的瀏覽器無法確定網站 SSL 憑證是由信任的憑證發證中心所核發時，開啟的 HTTPS 網頁會出現以下類似的警告訊息，提示使用者目前的連線傳輸狀態可能存在安全性風險，該步驟可有效降低使用者遭受中間人攻擊、防止敏感資訊外洩。



圖一：不安全的憑證警告訊息

貳、SSL 安全傳輸協定

HTTPS 的全文是 Hypertext Transfer Protocol over Secure Socket Layer，超文本傳輸安全協定，指的是安全的 HTTP 連接，為一種為原始網址模式加密防護的方法，現已為網路加密協定的全球化標準，基本適用於市面上大多數的瀏覽器[5][6]，以保密資料為目標研發，其安全基礎就是 SSL 協議，與 HTTP 最大的不同就是，在有 HTTPS 的網址輸入的資訊都是經由加密傳輸到網站伺服器，以杜絕在傳輸過程中被非法截取資料或篡改。基於其重要性，金管會即要求銀行組織必須使用 SSL 保護傳輸的敏感資訊，此外 HTTPS 亦已廣泛使用於交易支付和企業訊息系統中敏感訊息的傳輸。

SSL 目前已廣泛的應用在 HTTPS 連線上，可確保兩端點間資料傳輸的機密性和完整性及可驗證伺服器身分。在驗證伺服器身分方面，SSL Handshake 用來認證伺服器的身分，SSL 交握(SSL Handshake)的運作流程如圖二[7][8]。

1 : C ⟶ S : ClientHello
2 : S ⟶ C : ServerHello
: : [Certificate]
: : [ServerKeyExchange]
: : [CertificateRequest]
: : ServerHelloDone
3 : C ⟶ S : [Certificate]
: : ClientKeyExchange
: : [CertificateVerify]
: : ChangeCipherSpec
: : Finished
4 : S ⟶ C : ChangeCipherSpec
: : Finished

圖二：HTTPS Protocol (C 表示 Client，S 表示 Server)

中間人攻擊是一種很廣泛的攻擊手法，主要攻擊方式為找出網路協定中的漏洞，在兩通訊端點間插入一台虛擬主機偽裝成傳輸訊息的中繼點，再獨立連繫通訊者兩端，將自己偽裝成參與會話的終端，運用此種方式建立兩端的活動連結，使通訊者兩端誤以為是透過私密的方式直接對話，但實際上完整對話內容已被中間人所竊聽，並允許中間人攻擊者攔截通訊雙方的對話或插入新的惡意訊息，此方式最終還是能使通訊者接收到訊息，故難以被察覺，如下圖三所示。為有效防制此類攻擊，除了雙方資料傳輸必須加密外，亦必須驗證彼此的身分。藉由 SSL 交握的過程可讓用戶端和伺服器間安全地交換密鑰及雙方身分認證等相關規則，讓彼此進行資料傳輸時有所遵循。僅依靠加密是不夠的，客戶端必須驗證伺服器端的憑證，用來認證伺服器的身分，方可避免中間人攻擊。

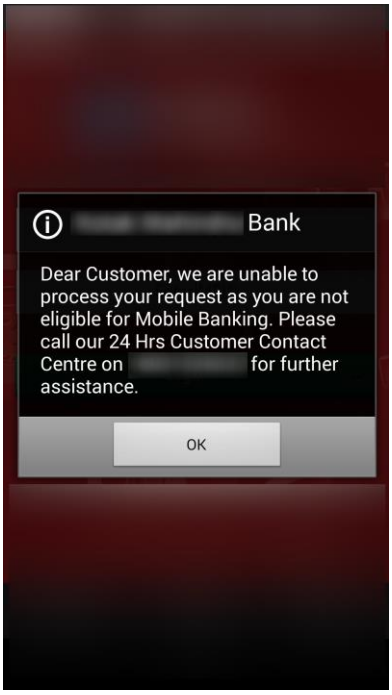
圖三：中間人攻擊

參、行動裝置應用程式漏洞

若組織自行開發應用程式，則程式開發人員必須確保安全傳輸加密協定(Protocol)中的每一個步驟完整且正確執行，否則即會造成資安上的漏洞。例如，於圖二的步驟二中，若伺服器傳給用戶端憑證，但用戶端並不驗證，將會造成中間人攻擊無法被 SSL 協定偵測到的風險，使得用戶和伺服器間傳送的敏感資訊外洩。在一般安全的情況下，若用戶遭受中間人攻擊時，瀏覽器會自動檢查伺服器端傳送過來的憑證有問題，逕自發出類似圖四、圖五的警告訊息，通知使用者目前連線的主機可能有安全上的風險。

圖四：電腦 Browser 示警圖

圖五：手機 Browser 示警圖



圖六：正確執行 SSL 之銀行手機 App

圖六顯示正確執行 SSL 之網路銀行手機 App，如圖可見，若嘗試以中間人的方式攔截訊息，應用程式會顯示錯誤訊息給用戶。若任何一個步驟未正確執行，則攻擊者可在用戶及伺服器均不知情的情況下，攔截所有機敏資訊。下為以 Android 平台為例，驗證漏洞的過程:

Android 上的應用程式均可下載並反編譯(decompile)，用戶很容易即可取得程式源碼。Android .apk（Application package）檔其實是 zip 格式的文件，在 Windows 作業系統下可以直接解壓縮，文件內容如表一。

表一：APK 文件內容

檔案名稱	說明
AndroidManifest.xml	程式配置文件
classes.dex	Dalvik 碼
resources.arsc	編譯後的資源文件
META-INF	certificate
res	資源檔案
assets	配置文件

我們利用解出來的檔案，再用 dex2jar 和 JD-GUI 兩個程式即可以把 APK 檔解開成 Java 原始檔，從而了解整個程式架構。我們為數個大型企業、銀行、政府組織進行自動化的 HTTPS 分析，發現有超過半數的組織的應用程式並沒有正確執行 HTTPS，圖七為我們發現漏洞的程式碼片段範例。

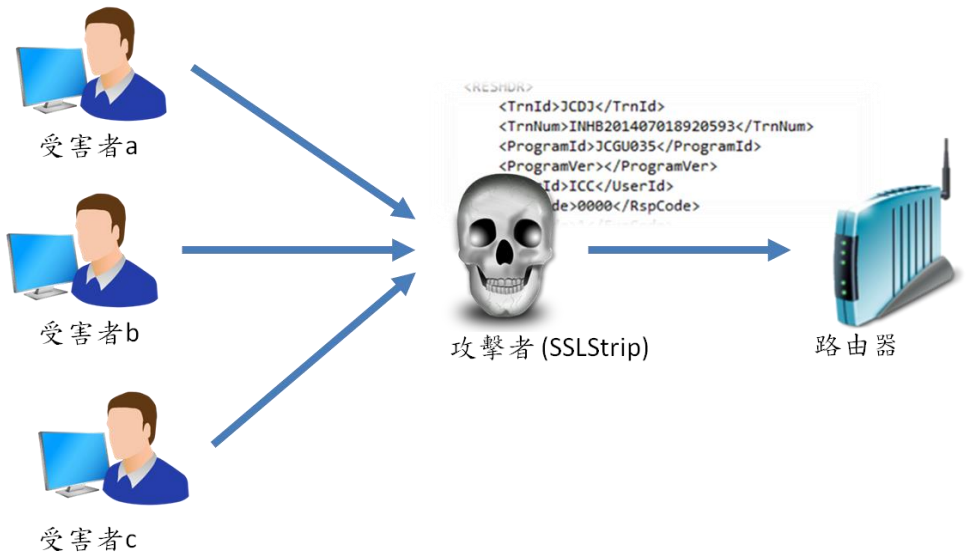
```
public boolean isClientTrusted(X509Certificate[]                    )
{
  return true;
}

public boolean isServerTrusted(X509Certificate[]                    )
{
  return true;
}
```

圖七：程式碼漏洞片段

此程式碼片段說明，於用戶端程式中，伺服器的憑證檢查直接回傳值 true，亦即忽略了驗證伺服器憑證的步驟。在這種情況下，即使有惡意攻擊者在中間攔截，用戶端也不會收到任何警告，攻擊者將可在受害人不知情的情況下，攔截並取得敏感資訊，從而可能入侵受害人的網路銀行、電子信箱等等。

如圖八所示，攻擊者可利用 Arpspoof 結合此類漏洞執行中間人攻擊，如此一來，攻擊者可成為中繼點，使用者原本要發送給目標伺服器的資料，就會先通過攻擊者截持的主機，這時攻擊者就可以監聽甚至修改數據，可輕易取得許多用戶的帳號密碼敏感資訊，甚至也可能包括信用卡的資料，如圖九所示。



圖八：攻擊模型

圖九：中間人攻擊攔截範例

此處舉的例子僅為沒有正確實現 SSL 之其一範例，目前有非常大量的隨身裝置應用程式有此類問題，尚待解決。其他諸多協定(protocol)，亦有執行不完整的問題，可造成機敏資訊外洩、網路銀行被入侵、重大應用程式被非法登入等等。

肆、結論

隨著電子通訊發展快速，智慧型手機已成為每個人生活中必要配備，而多數 App 應用程式卻隱藏著重大的資安風險，讓使用者在不知情的狀況下，導致個人機密資料外洩。

面對手機 App 應用程式如雨後春筍的大量崛起，再加上資安的相關議題日益受到重視，有關單位在制定相關規範時，應有完善的技術稽核機制，確認並正確地執行資安程序，而非僅流於表面形式。根據本文所探討的安全性漏洞而言，儘管手機 App 應用程式表面上已使用 SSL 加密(使用 HTTPS)，實則並未完整地正確執行其安全傳輸加密協定(protocol)，進而造成用戶有嚴重個資外洩的風險。

我們已聯繫通報主要手機生產商相關漏洞，並希望手機 App 應用程式於上架前能針對資安項目進行嚴謹審查，除了保障使用者的權益，亦能維護手機 App 應用平台的品質，對於開發廠商與消費者不啻為好事一件。

開發廠商針對自行開發或外包的手機 App 應用程式，應建立一套資安規範與工具進行檢核，我們也建議相關單位在應用程式委託外包之際，需訂立相關標準法規，並提出檢測標準及工具，以避免嚴重資安事件的發生。此外，政府有關單位應訂定完善的行業認證標準和相關法律規定，並制定應用平台與手機應用程式安全標準，以確保應用程式開發絕對符合資安標準規範，並遵守開發應用程式安全相關準則。

手機應用程式相關的漏洞為數眾多，此篇文章以 HTTPS 為例，希望藉此案例提醒相關人員引以為戒，正視此類問題的普遍性及嚴重性，有效防止惡意攻擊者運用漏洞進行攻擊、竊取使用者機敏資料。相關研究人員亦基於手機安全檢測經驗和理論基礎，針對手機 App 應用程式進行多種協定驗證，發現此類型問題非常嚴重且影響深遠，亟需訂立稽核措施和改善。

參考文獻

[1] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin and D. Wagner. "Android permissions: User attention, comprehension, and behavior," *Proceedings of the Eighth Symposium on Usable Privacy and Security*, 2012.

[2] E. Lawrence, "HTTPS Security Improvements in Internet Explorer 7," *MSDN. 31*, January 2006.

[3] "Mozilla Firefox Privacy Policy," *Mozilla Foundation*, 27 April 2009.

[4] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov, "The most dangerous code in the world: validating SSL certificates in non-browser software, " *Proceedings of the 2012 ACM conference on Computer and communications security.*

[5] "SSL/TLS in Detail," *Microsoft TechNet*, http://technet.microsoft.com/en-us/library/cc785811.aspx.

[6] "Description of the Secure Sockets Layer (SSL) Handshake," *Support Microsoft*, http://support.microsoft.com/kb/257591.

[7] "HTTP over TLS," http://www.ietf.org/rfc/rfc2818.txt, 2000.

[8] "Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile," http://tools.ietf.org/html/rfc5280, 2008.

[9] J. Claessens, V. Dem, D. D. Cock, B. Preneel, J. Vandewalle, "On the Security of Today's Online Electronic Banking Systems," *Computers & Security*, Vol 21, Issue 3, 2002.

[10] P. Eckersley and J. Burns, "An observatory for the SSLiverse," *In DEFCON*, 2010.

[11] "The Secure Sockets Layer (SSL) protocol version 3.0", http://tools.ietf.org/html/rfc6101, 2011.