



Research Center for information Technology Innovation

**TWISC**

TAIWAN INFORMATION SECURITY CENTER

中央研究院-資通安全研究與教育中心

資安前瞻關鍵技術基礎研究計畫

手機資料異常行為分析相關研究

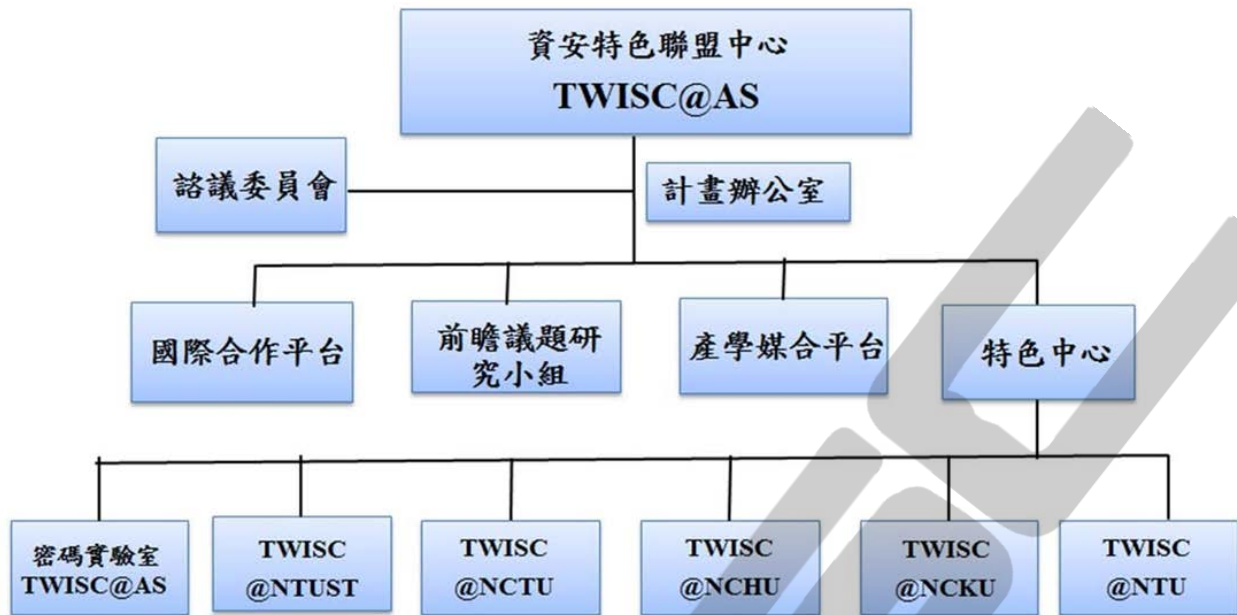
The Security Detection of Android Behavior



前瞻議題研究小組 張宏昌 博士

中華民國 106 年 11 月

## 資安特色中心暨聯盟



資安特色中心暨聯盟，是行政院為確立國土數位安全，加速數位經濟發展而執行的一項計畫。成立目的：為政府、業界積極培育資安專業人才，發展前瞻資安科技。中央研究院資創中心執行「資安特色中心暨聯盟」計畫，結合大專院校資訊安全學有專長的教授及研究人員，成立五大資安特色中心，特色中心各自有專業的分工。本計畫將延續過去十年資通安全研究與教育中心（TWISC）與國際著名資安中心合作關係為基礎，積極拓展實質資安研究的國際合作網絡，引進國外尖端資安技術，促使國內的資安科技研發持續成長，達成關鍵技術自主研發為目標；透過產學媒合平台的建立，將學界的研發成果，擴散至產業界。並設立前瞻議題研究小組，掌握資安研究趨勢與新興技術所前在的資安威脅，為政府及業界提供專業的資安諮詢，並培養全方位的資安人才，提升我國資安防護技術水準。本計畫預計執行三年，第一年度（106 年度）重點工作為成立建置資安特色聯盟中心，並且規劃諮議委員會及計畫辦公室，並且建置『國際合作平台』、『前瞻議題研究小組』、『產學媒合平台』與『特色中心』；其主要工作內容為建置資安特色聯盟中心，集結國內資安技術與人才資料庫，並將所

有相關人才的技術整合一個開放式產學媒合平台，方便媒合企業們尋找相關人才與技術。第二年工作重點在『資安特色聯盟中心』之相關技術建置完善，其中包含國際合作、前瞻議題資訊及產學媒合平台，提供最新資安技術與人才資訊，逐步技轉產業界或者人才就業，協助產業提昇自我資通安全防禦之研發能力。第三年（108 年度）之重點工作重點，擴散『資安特色聯盟中心』的成果，預計將提供國際或國內資安相關技術及人才培育資訊名單，開放企業技術移轉或者產學合作及人才媒合等，將其研究成果技轉法人團體與業界，提升其資訊安全資訊能量。

# 10 月重大資安事件

## SWIFT 遭駭事件大回顧

儘管遠東銀行 SWIFT 遭駭事件是臺灣首例，也創下臺灣銀行遭駭盜轉金額的新記錄，不過，這並非是單一事件，從 2013 年迄今，除了遠東銀行事件之外，全球已有 6 起銀行因 SWIFT 系統被駭客組織入侵，造成銀行資金損失的事件發生。其中，去年最廣為人知的孟加拉中央銀行被駭客盜領 8,100 萬美元的事件，也是類似的案例，在孟加拉中央銀行的事件曝光後，先前遭到類似手法攻擊的事件，也一一浮上檯面，包括 2013 年孟加拉的索納利銀行、2015 年 1 月厄瓜多銀行、10 月菲律賓銀行，以及 12 月越南先鋒銀行的 SWIFT 系統，都遭到類似的手法攻擊，駭客都是先獲得目標銀行的 SWIFT 權限，再向其他銀行提出轉帳指令，最後，駭客還會將相關的犯罪記錄刪除，讓追查犯罪的過程更加困難。

全球金融機構通用的 SWIFT 通訊系統，是銀行間交易中交換結構化的電子訊息的平臺，處理支付和交易結算等商業流程，駭客組織可能利用 SWIFT 系統的漏洞，或是網路進行攻擊，來取得 SWIFT 系統的操作權限，植入惡意程式，將該銀行的資金轉到偽造的帳戶。攻擊銀行轉帳交易系統 SWIFT 的系統事件，最早出現在 2013 年，當時孟加拉國的索納莉銀行（Sonali Bank）遭到駭客攻擊，成功盜領該銀行 25 萬美元的資金，索納莉銀行表示，駭客透過網路在銀行內部一臺電腦植入惡意程式 keylogger，竊取相關的系統帳號和密碼，取得操作權限，並使用 SWIFT 系統傳送偽造的轉帳申請。

SWIFT 組織未深入解說案件詳情，僅說明未來攻擊事件會持續增加，同時會增加更多強制性安全措施要求會員銀行遵守，以避開駭客攻擊，同時，SWIFT 也要求會員銀行遵守 16 項強制控制措施，並將自 2018 年起對會員銀行進行稽核，未遵循規定的金融機構合作夥伴，以及其地金融監管機關，將被告知其未遵循狀況。

### SWIFT 系統遭駭銀行清單

※2013 年：索納利銀行

※2015 年 1 月：厄瓜多銀行

※2015 年 10 月：菲律賓一家銀行

※2015 年 12 月 8 日：越南先鋒銀行

※2016 年 2 月 5 日：孟加拉國央行 (Bangladesh Central Bank)

※2016 年 6 月：烏克蘭多家銀行

※2017 年 10 月 5 日：臺灣遠東商業銀行

\*參考資料:iThome

## 摘要

現今智慧型手機技術所帶來的方便性，使得越來越多人使用智慧型手機，然而惡意程式數量不斷提升、手機廠商私自竊取用戶資料、手機個資外洩等新聞日出不窮，儼然手機資訊安全已經成為資安上的一大問題。國內學者對於行動設備在惡意程式檢測方面提出許多方法，但並沒有針對實體手機內建軟體進行相關研究檢測及分析。本研究提出一套動態分析系統，主要是來觀察手機使用狀態及異常特徵，實驗首先針對惡意程式進行動態分析以及靜態分析，將惡意程式所獲得的特徵來檢測實體手機是否與惡意程式特徵相似。在本研究的實驗結果中發現實體手機都會傳輸敏感資料，而靜態分析中 MIUI 手機較其他廠牌有較大的安全疑慮。

## 背景及動機

隨著科技進步與行動網路 3G、4G 的提升，手機不在限制於使用簡訊、撥打電話等功能，使用者對於手機的要求不外乎速度快、能夠瀏覽網頁、收發 e-mail、運動記錄、線上看影片等應用程式，顯然手機漸漸成為不可或缺的生活必需品，因此各大廠商紛紛投入智慧型手機平台市場，開發了許多應用軟體來滿足使用者的需求。

根據 IDC 所公布的智慧型手機市占率如表 1，從 2012 年至 2015 手機作業系統的市占率來看主要還是以 Google 公司所開發的 Android 最高目前市占率已經達到飽和持平的狀況，第二為 Apple 公司所開發的 iOS 系統可以看出 2015 年與 2014 年市占率有提升 3.1%，第三為 Microsoft 公司所開發的 Windows Phone，第四為 BlackBerry 開發的 BlackBerry OS，由表 1 可以發現 BlackBerry 的市占率越來越低到 2015 年甚

至降到 0.3%，因此我們可以知道在智慧型手機市場上 Android 為全球最多人使用的作業系統。

表 1 Smartphone OS Market Share, Q1 2015

Period	Android	iOS	Windows Phone	BlackBerry OS	Others
Q1 2015	78.0%	18.3%	2.7%	0.3%	0.7%
Q1 2014	81.2%	15.2%	2.5%	0.5%	0.7%
Q1 2013	75.5%	16.9%	3.2%	2.9%	1.5%
Q1 2012	59.2%	22.9%	2.0%	6.3%	9.5%

隨著智慧型手機市場快速的成長，相關的資訊問題也越來越受到重視，近年來 Android 系統上的除了惡意程式不斷的快速成長外，也發現在 Android 手機裝置上之系統或應用程式本身就存在著漏洞，有些手機廠商在內建應用程式會竊取手機上的隱私資訊等行為傳輸給廠商主機等違法行為，在各大新聞中播放了一些關於手機資安報導，例如：MIUI 未經過使用者的同意，將手機資料偷傳資料到北京、趨勢科技也發現一些 Android 內建軟體的漏洞會讓使用者個資陷入危險或被利用來發動攻擊、國家通訊傳播委員會(NCC)近日針對市面上 12 款熱門手機進行檢測包含有 iPhone 的蘋果、HTC 的宏達電、三星及華碩等，檢測結果為「一經使用，就會自動傳送訊號到製造商的伺服器去註冊，會有資安疑慮」、Sony 手機儲存空間出現了一個名為「baidu」的資料夾，這個「baidu」資料夾是無法被刪去，且「baidu」資料夾會透過 Sony 預載程式 MyXperia 自動連接至中國伺服器等報導。

顯然智慧型手機在資訊安全上已成為新興消費者保護課題，因此我們得知除了惡意程式本身的威脅，手機內建應用程式也有資訊安全的疑慮存在。

## 研究方法與架構

本研究主要是針對實體手機本身內建應用程式進行監控，將相關研究中所得出的惡意程式特徵結果統整提出一套檢測方法使用在 Android 實體手機上。首先將實體手機還原為原廠設定進行檢測，在靜態分析中將手機內部應用程式取出來做特徵碼掃描以及權限分析，在動態分析中檢測手機內部應用程式的使用狀況 CPU、RAM、Logcat 訊息、網路封包，將取得結果與惡意程式的特徵進行分析是否有安全疑慮，圖 1 為系統架構。

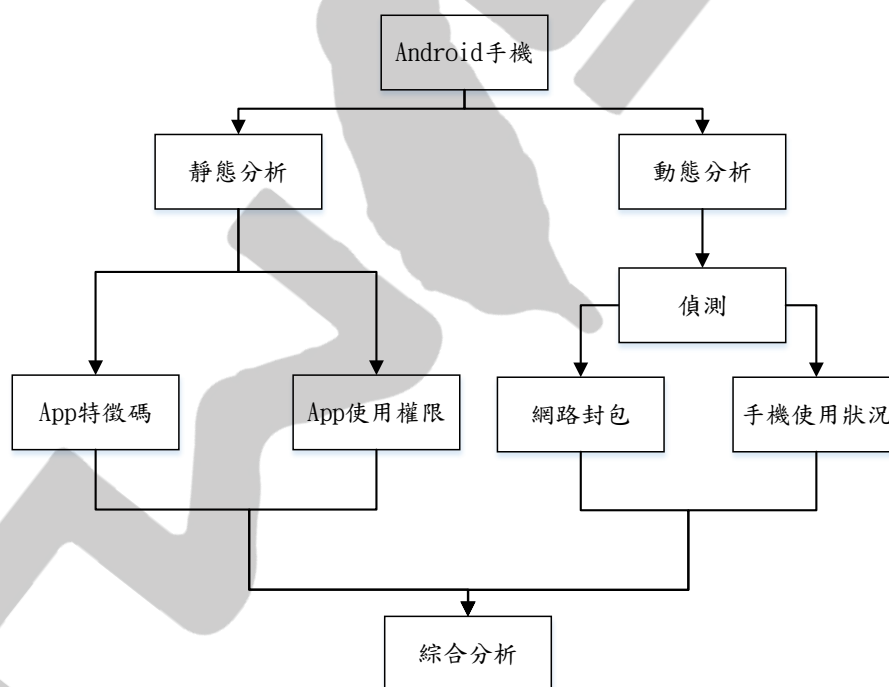


圖 1 系統架構

目前手機動態分析工具都以虛擬機為主，而本研究要以實體手機作為實驗，因此撰



寫一個系統要取得手機使用的狀態此系統為 ADB 動態系統。此系統主要包含應用程式網路行為、CPU 使用狀況、RAM 的使用狀況、Logcat 訊息等。

本研究自行撰寫的 ADB 動態系統使用 Android SDK tools ADB 的指令並搭配 C#撰寫整合系統所需功能，再將手機內部資料取出儲存於.csv 格式方便後續分析使用。表 2 為此系統功能。

表 2 系統功能

功能	說明
行動裝置資訊	了解廠牌、型號、Android 版本、目前手機時間
電池	了解目前手機電池電量、溫度
基本資料	了解手機韌體版本、裝置機板代號等資訊
儲存空間資訊	了解手機目前資料夾使用容量
網路介面	了解目前手機內部 Wi-Fi、p2p 等網路介面
網路連線	了解目前手機應用程式使用網路狀態、IP
CPU 狀態	了解目前手機 CPU 頻率、CPU 溫度
CPU 使用資訊	了解目前手機系統使用狀況
RAM 使用資訊	了解目前手機記憶體使用狀況
程序使用資訊	了解目前手機 CPU 使用狀況
LOG 訊息	了解目前手機調用狀況

使用表 2 的功能我們需要使用 ADB 工具讓手機與系統溝通，並且在系統中互相調用資料，例如網路連線使用 ADB 命令可以得知網路使用狀況，但無法得知是哪個應用程式所使用的必須再搭配其他 ADB 命令，因此本系統使用多個 ADB 命令。表 3 列出幾個系統較為重要的命令。

表 3 ADB 指令說明

指令	功能
<b>adb shell dumpsys package p</b>	手機應用程式的 UID 碼
<b>adb shell dumpsys battery</b>	手機電池的使用狀況
<b>adb shell dumpsys meminfo</b>	手機應用程式使用的記憶體的狀況
<b>adb shell df</b>	手機資料夾使用狀況
<b>adb shell getprop</b>	手機硬體、廠商、IMEI 等資訊
<b>adb shell logcat</b>	手機調用資訊及狀況
<b>adb shell netcfg</b>	手機網路使用狀況
<b>adb shell top</b>	手機 CPU 使用狀況

圖 2 為主系統畫面，此系統操作容易、方便，先將手機與電腦連線再按下連線按鈕即可使用，而且此系統不論是哪一 Android 手機都可以使用，圖 3、圖 4 為系統使用畫面。

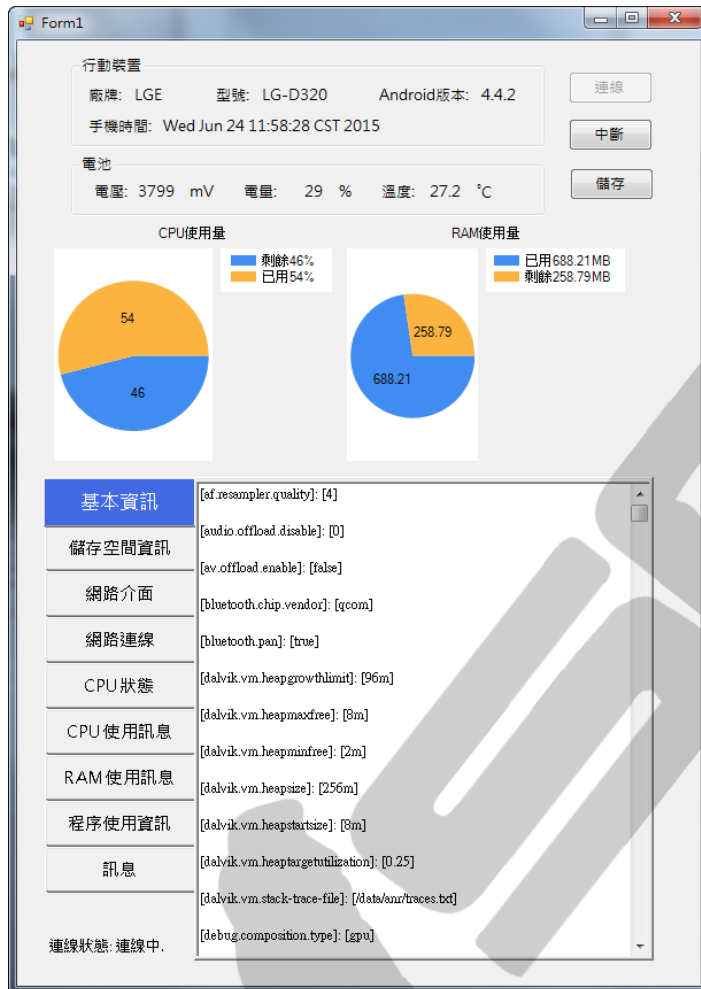


圖 2 主系統畫面



圖 3 網路連線

基本資訊	包名:system_server PID:841
儲存空間資訊	使用率:85% 用戶使用率:53% 系統使用率:31%
網路介面	包名:com.nextmediatv PID:14792
網路連線	使用率:14% 用戶使用率:14% 系統使用率:0%
CPU 狀態	包名:adbd PID:534
CPU 使用訊息	使用率:7% 用戶使用率:0.6% 系統使用率:6.3%
RAM 使用訊息	包名:kworker PID:24487
程序使用資訊	使用率:1.3% 用戶使用率:0% 系統使用率:1.3%
訊息	包名:com.supercleaner PID:23584
連線狀態: 連線中.	

圖 4 CPU 使用訊息

## 靜態分析

靜態分析為特徵碼分析以及權限分析，特徵碼分析本研究使用 VirusTotal，是一個網路上知名的掃毒服務網站，VirusTotal 整合了小紅傘(Avira)、AVG、ClamAV、NOD32、F-Secure、GData、卡巴斯基、McAfee、Panda、BitDefender、趨勢科技與 Norton Antivirus 等 47 種全世界最優秀的防毒軟體，VirusTotal 可以給予相當詳細的分析。圖 5 為 VirusTotal 掃描結果。

virustotal

SHA256: 7bb9b0b3ca397078957bc63541347cb904a74a1139f07466aef800eaa0339abc

檔案名稱: 5F54F8C613AAED33F12381B3F4F9B2AB.apk

偵測率: 36 / 56

分析日期: 2015-05-06 01:51:37 UTC (1 月, 2 週前)

圖 5 VirusTotal 掃描結果

權限分析要判別應用程式(APP)是否有安全疑慮，因此我們將先前有關惡意程式權限分析的研究選出幾個可能造成手機安全疑慮的權限，再加上此研究主要了解是否有手機本身將個人資料外洩的疑慮，多加了可讀取手機 LOG 權限(如 READ\_CALL\_LOG、READ\_LOG)進行分析。表 4 為此研究所列出來有安全疑慮的權限。

表 4 權限瀏覽

權限名稱	權限功能
ACCESS_COARSE_LOCATION	允許程式存取 Wi-Fi 網路狀態 訊息
ACCESS_FINE_LOCATION	允許程式存取精確位置(如 GPS)
CALL_PHONE	允許一個程式初始化一個電 話撥號不需透過撥號使用者 界面需要使用者確認
CALL_PRIVILEGED	允許一個程式初始化一個電 話撥號不需透過撥號使用者 界面需要使用者確認
CAMERA	允許存取使用照相裝置
DEVICE_POWER	允許程式開啟或關閉手機
DOWNLOAD_WITHOUT_NOTIFICATION	允許程式可不告知使用者， 自行下載東西
GET_ACCOUNTS	允許程式取得帳戶清單
INSTALL_PACKAGES	允許程式下載應用程式
MANAGE_ACCOUNTS	允許程式執行新增、移除帳 戶和刪除帳戶密碼等作業

MODIFY_PHONE_STATE	允許程式控制裝置的電話功能。擁有這項權限的應用程式可在未通知您的情況下，任意切換網路、開啟或關閉手機無線電等
READ_CALL_LOG	允許程式讀取通話紀錄
READ_CONTACTS	允許程式讀取聯絡人資料，包括您與特定聯絡人通話、傳送電子郵件或使用其他通訊方式的互動頻率
READ_LOGS	允許程式讀取系統的各种記錄檔，可能包含個人或私人資訊
READ_PROFILE	允許程式讀取裝置上儲存的個人資料，例如您的姓名和聯絡資訊
READ_SMS	允許程式讀取簡訊
REBOOT	允許程式能夠重新啟動裝置
RECEIVE_SMS	允許程式接收和處理簡訊。這項設定可讓程式監控傳送

	至您裝置的訊息，或在您閱讀訊息前擅自刪除訊息
SEND_SMS	允許程式發送簡訊
SEND_SMS_NO_CONFIRMATION	允許在沒有使用者的允許下發出訊息
WRITE_APN_SETTINGS	允許程式變更網路設定，並攔截及檢查所有網路流量，惡意程式可能利用此功能，在您不知情的情況下監控、重新導向或修改網路封包
WRITE_CONTACTS	允許程式修改裝置儲存的聯絡人資料，包括您與個別聯絡人通話、電郵或以其他通訊方式聯絡的頻率。這項權限允許應用程式刪除聯絡人資料
WRITE_PROFILE	允許程式新增或變更裝置上儲存的個人資料，例如您的姓名和聯絡資訊。這項設定可讓應用程式識別您的身

	分，並可能將您的個人資料傳送給他人
WRITE_SECURE_SETTINGS	允許程式修改系統安全設定資料
WRITE_SMS	允許程式寫入裝置或 SIM 卡中儲存的短訊。惡意應用程式可能會藉此刪除您的訊息

靜態分析使用的工具為 Virustotal 以及 Android SDK tools AAPT。靜態分析流程主要先將手機回到原廠設定將手機內建應用程式取出，使用 Virustotal 檢測應用程式特徵碼是否有安全疑慮，以及使用 AAPT 進行應用程式權限掃描如圖 6，再經由上述列出可能有安全疑慮的權限進行篩選，將有安全疑慮的應用程式記錄下來如圖 7 為靜態分析流程。

```
D:\Android\sdk\platform-tools>aapt d permissions CameraCommon.apk
package: com.sonymobile.cameracommon
uses-permission: android.permission.CAMERA
uses-permission: com.sonyericsson.permission.CAMERA_EXTENDED
uses-permission: com.sonymobile.permission.CAMERA_ADDON
uses-permission: com.sonymobile.sonyselect.SHOW_CUSTOM_CONTENT
uses-permission: android.permission.WRITE_SETTINGS
uses-permission: android.permission.WRITE_SECURE_SETTINGS
```

圖 6 AAPT 抓取權限



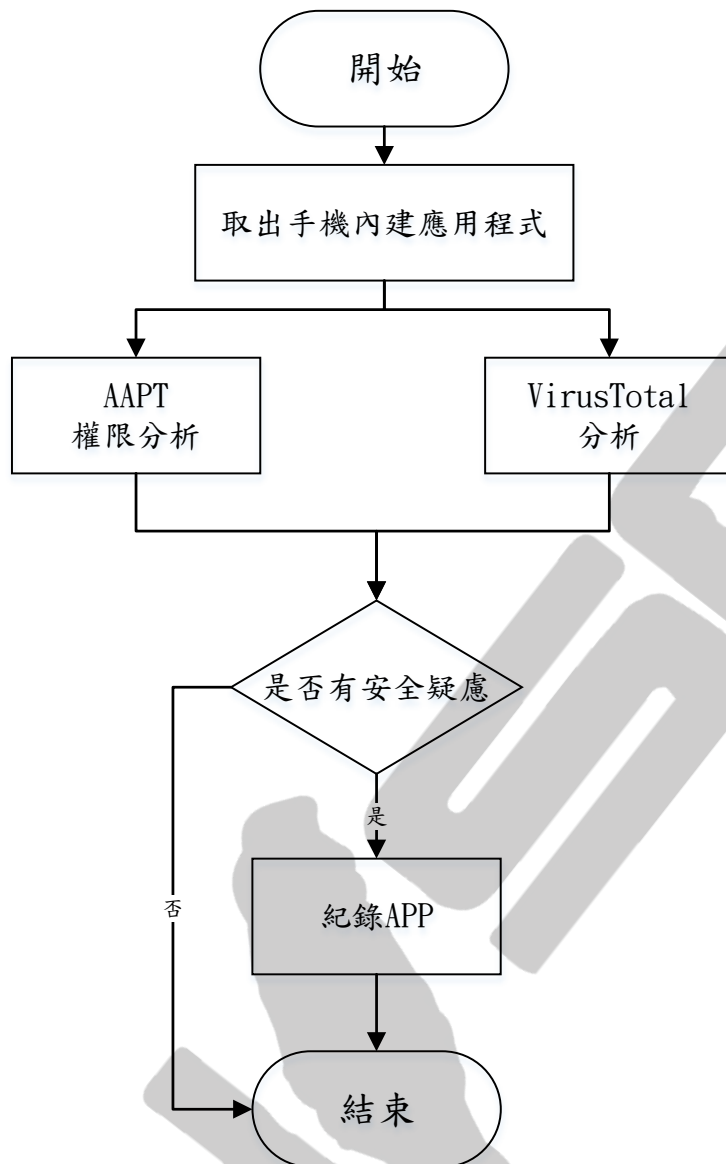


圖 6 靜態分析流程

## 動態分析

動態分析使用的工具為 TCPDUMP 以及本實作系統 ADB 動態系統。TCPDUMP 主要是能夠抓手機網路封包，此實驗過濾 HTTP-POST 以及 HTTP-GET 查看傳輸內容是否有敏感隱私資料的行為，因網路封包抓取需要實體手機執行才能夠看得出封包的使用狀況加上不同廠牌的手機所顯示出的內容也不一樣，因此在此研究列為動態分析。ADB 動態系統為手機執行時觀察手機使用狀況 CPU、RAM、Logcat 訊息將手機狀況記錄下來後續進行分析。

動態分析流程為圖 8，先將手機回到原廠設定後開啟手機，使用 ADB 動態系統以及 TCPDUMP 開始監控手機的動作，透過 TCPDUMP 抓取網路封包主要是針對手機啟動(開機時)所傳送的資訊以及手機內建軟體進行檢測，同時系統記錄手機目前應用程式使用的網路、CPU、RAM 狀態，再將資料彙整後續分析是否實體手機是否再使用是有安全疑慮。

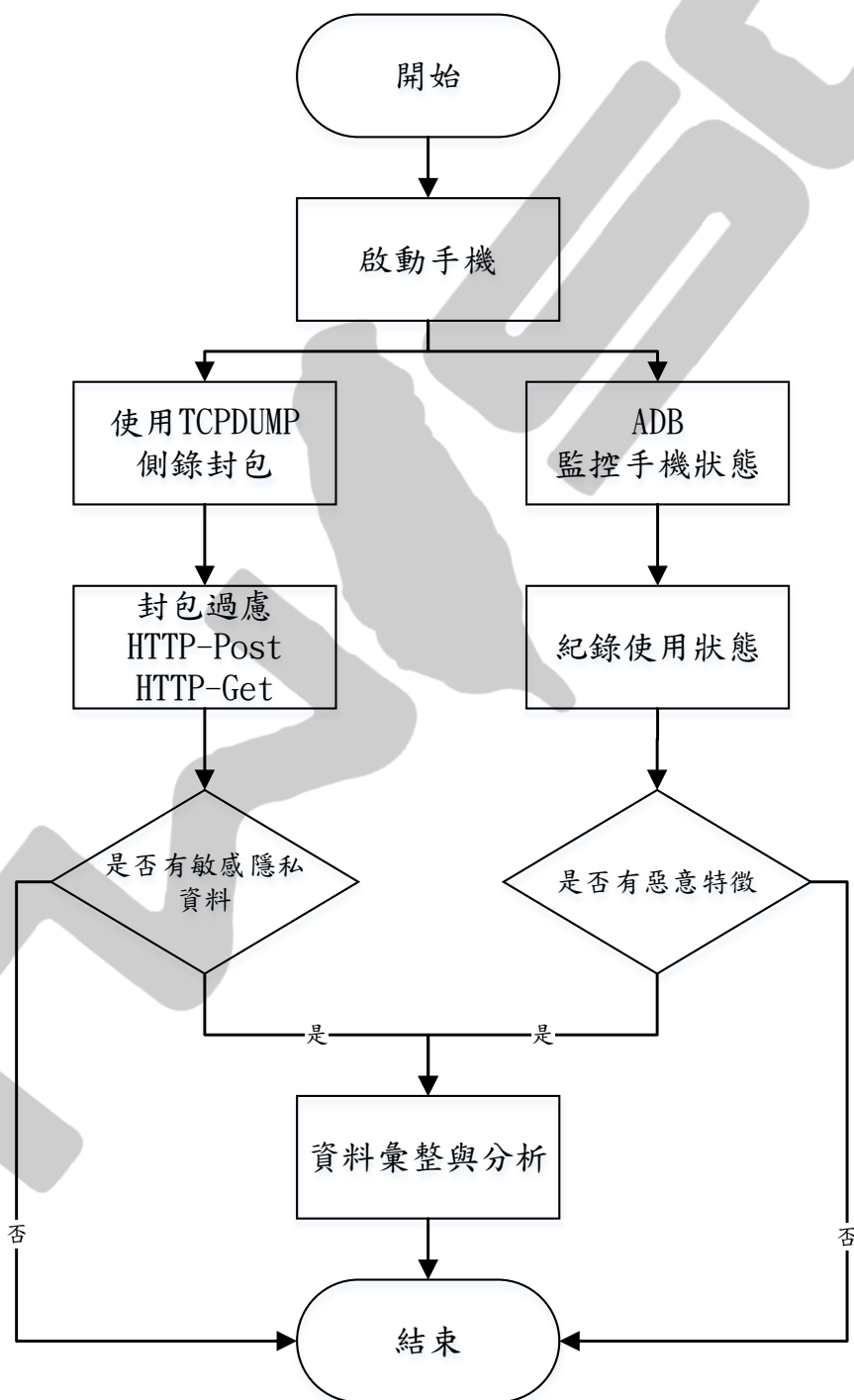


圖 8 動態分析流程

# 測試結果

本實驗選擇四台手機 HTC、MIUI(小米)、Sony、LG 作為實驗手機，經由靜態分析以及兩個小時的動態分析，監控手機是否有安全疑慮。

## 靜態分析

圖 9 可以看到經由 VirusTotal 掃描結果只有 MIUI 88 個裡面有 3 個 APK(GuardProvider.apk、SmsReg.apk、Provision.apk)被掃出有安全疑慮的可能如表 5、表 6、表 7，剩下的 HTC、Sony、LG 掃描出的結果皆為安全。

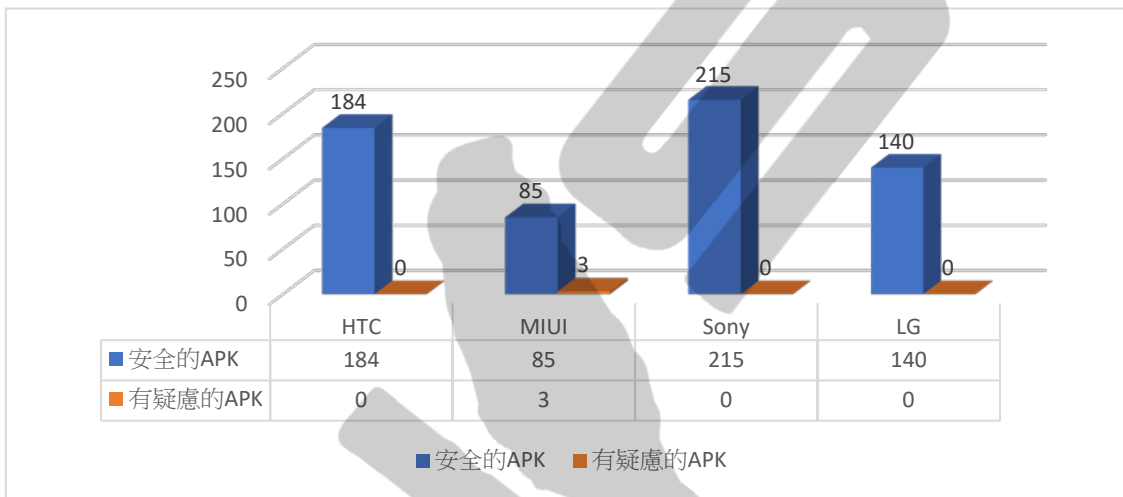


圖 9 VirusTotal 掃描結果

表 5 GuardProvider.apk 掃描結果

名稱	GuardProvider.apk	
偵測率	7/55	
結果	檢測軟體	結果
	AVG	Android/Deng.HOT
	Avast	Android:SMSreg-BBX [PUP]
	Avira	SPR/ANDR.SmsReg.AM.Gen
	Baidu-International	Hacktool.Android.SMSreg.GD

	ESET-NOD32	a variant of Android/SMSreg.GD potentially unsafe
	Ikarus	AdWare.AndroidOS.SMSreg
	TrendMicro-HouseCall	Suspicious_GEN.F47V0428

表 6 SmsReg.apk 掃描結果

名稱	SmsReg.apk	
偵測率	7/57	
結果	檢測軟體	結果
	Avast	Java:Malware-gen [Trj]
	F-Secure	Riskware:Android/SmsReg
	McAfee	Artemis!32901434AAC7
	McAfee-GW-Edition	Artemis
	Sophos	Andr/Gedma-A
	TrendMicro-HouseCall	Suspicious_GEN.F47V030 9
	TotalDefense	Tnega.XAYU!suspicious

表 7 Provision.apk 掃描結果

名稱	Provision.apk	
偵測率	1/56	
結果	檢測軟體	結果
	Alibaba	A.L.Rog.Provision

在每台手機中相同功能的 APK 所要求的權限會有所不同，將每台手機 APK 可能有安全疑慮的權限結果展開出來看，發現 MIUI 要求的權限與其他品牌比起來要求更多有

安全疑慮的權限，尤其個人隱私所需要的權限如圖 10 MIUI 要求的都比其他手機來的多。

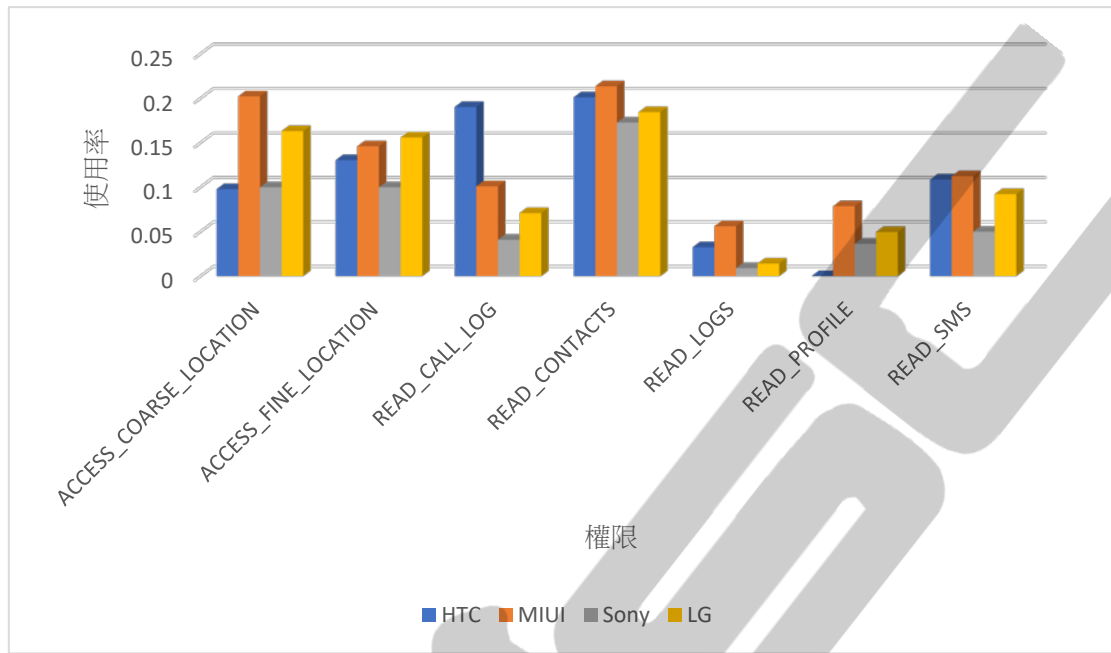


圖 10 個人隱私所需權限分析表

將 25 個有安全疑慮的權限統整出來的權限使用率，我們發現 MIUI 高達 13 個次數最高；HTC 有 6 個最高；LG 有 6 個最高；Sony 有 0 個使用率最高如表 21。

表 8 25 個有安全疑慮的權限使用率的手機廠牌

手機廠牌	有安全疑慮的權限
HTC	CALL_PHONE、INSTALL_PACKAGES、 READ_CALL_LOG、READ_CONTACTS、 WRITE_CONTACTS、WRITE_SMS
MIUI	ACCESS_COARSE_LOCATION、 CAMERA、 DOWNLOAD_WITHOUT_NOTIFICATION、 MANAGE_ACCOUNTS、 MODIFY_PHONE_STATE、READ_LOGS、 READ_PROFILE、 READ_SMS、SEND_SMS、 SEND_SMS_NO_CONFIRMATION、 WRITE_APN_SETTINGS、WRITE_PROFILE、 WRITE_SECURE_SETTINGS
LG	ACCESS_FINE_LOCATION、 CALL_PRIVILEGED、DEVICE_POWER、 GET_ACCOUNTS、REBOOT、RECEIVE_SMS

## 動態分析

在靜態分析我們得知 MIUI 手機內部有三個可能有安全疑慮的 APK (com.miui.guardprovider、com.mediatek.smsreg、com.android.provision)，在動態分析時我們取手機在兩個小時內 CPU、RAM 的動作如圖 10、圖 11，我們發現這三個 APK 在開機時會被啟動，com.mediatek.smsreg、com.android.provision 啟動後過幾分鐘後就沒有動靜了，而在 com.miui.guardprovider 方面發現他會定期在每半個小時會有動作，而在 RAM 的部份變動則不高。

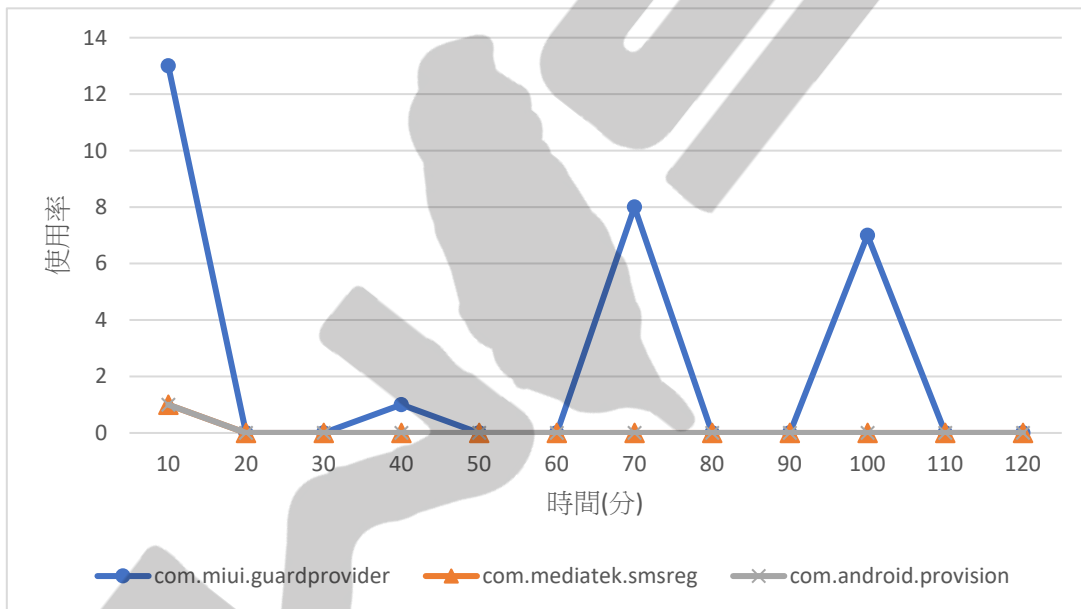


圖 10 MIUI 有安全疑慮 APK 的 CPU 使用率

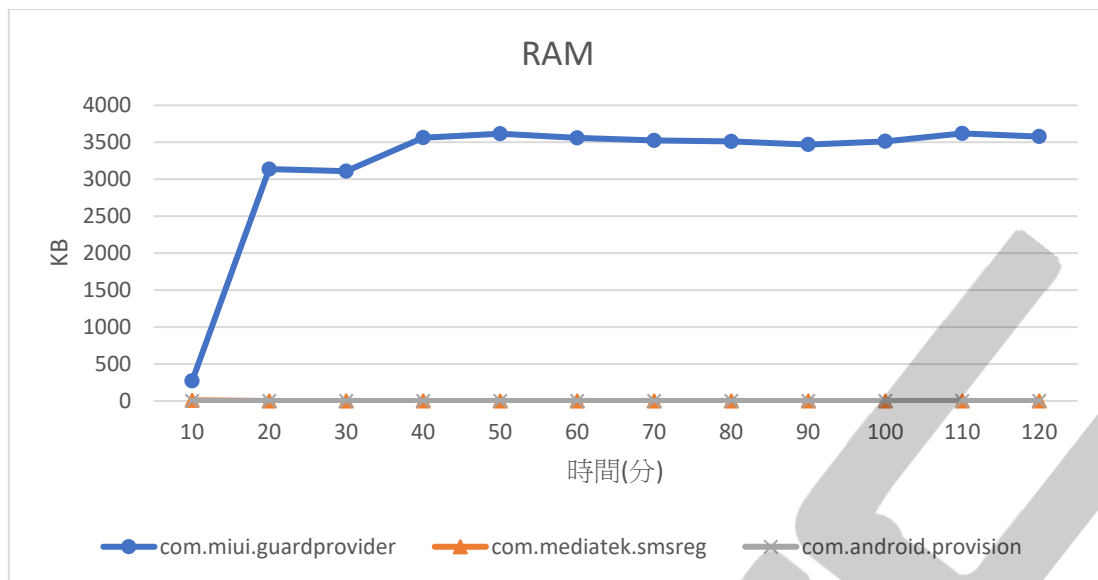


圖 11 MIUI 有安全疑慮 APK 的 RAM 使用狀況

查看一下 Logcat 的紀錄發現 com.miui.guardprovide 主要先連線主機，調用手機內部資源的內容再經由網路傳輸出去，主要傳輸內容為使用者的資料，封包主要內傳輸手機設備等資訊。

```

05-13 16:44:04.237 I/FunctionHandler( 3360): resources.arsc
05-13 16:44:04.237 I/FunctionHandler( 3360): res/menu/activity_main.xml
05-13 16:44:04.237 I/FunctionHandler( 3360): res/layout/activity_main.xml
05-13 16:44:04.236 I/FunctionHandler( 3360): res/drawable-xhdpi/ic_launcher.png
05-13 16:44:04.236 I/FunctionHandler( 3360): res/drawable-xhdpi/ic_action_search.png
05-13 16:44:04.224 I/FunctionHandler( 3360): lib/armebabi-v7a/libmsparser-1.0.0.so
05-13 16:44:04.210 I/FunctionHandler( 3360): lib/armebabi-v7a/libsmschecker-1.0.1.so
05-13 16:44:04.197 I/FunctionHandler( 3360): lib/armebabi-v7a/libdexanalysisd.so
05-13 16:44:04.185 I/FunctionHandler( 3360): lib/armebabi-v7a/libdexanalysis.so
05-13 16:44:04.181 I/FunctionHandler( 3360): lib/armebabi-v7a/libcryptor-1.0.0-legacy.so
05-13 16:44:04.148 I/FunctionHandler( 3360): lib/armebabi-v7a/libams-1.1.0-legacy.so
05-13 16:44:04.147 I/FunctionHandler( 3360): com/qq/jce/wup/wup.properties
05-13 16:44:04.031 I/FunctionHandler( 3360): classes.dex
05-13 16:44:04.028 I/FunctionHandler( 3360): assets/ye_public_phone
05-13 16:44:04.018 I/FunctionHandler( 3360): assets/ye_phone_tag
05-13 16:44:04.018 I/FunctionHandler( 3360): assets/ye_license
05-13 16:44:04.017 I/FunctionHandler( 3360): assets/ye_intl_phone
05-13 16:44:04.017 I/FunctionHandler( 3360): assets/ye_fixed_phone
05-13 16:44:03.989 I/FunctionHandler( 3360): assets/ye_base.ldb
05-13 16:44:03.985 I/FunctionHandler( 3360): assets/rule_store.sys
05-13 16:44:03.951 I/FunctionHandler( 3360): assets/qv_base.amf
05-13 16:44:03.883 I/FunctionHandler( 3360): assets/model.db
05-13 16:44:03.874 I/FunctionHandler( 3360): assets/model
05-13 16:44:03.874 I/FunctionHandler( 3360): assets/licence.conf
05-13 16:44:03.873 I/FunctionHandler( 3360): assets/fonts/adddetector.ttf
05-13 16:44:03.873 I/FunctionHandler( 3360): assets/antispam_profiles.db
05-13 16:44:03.873 I/FunctionHandler( 3360): YhdsEngine-version
05-13 16:44:03.872 I/FunctionHandler( 3360): AndroidManifest.xml
05-13 16:44:03.872 I/FunctionHandler( 3360): META-INF/CERT.RSA
05-13 16:44:03.871 I/FunctionHandler( 3360): META-INF/CERT.SF
05-13 16:44:03.871 I/FunctionHandler( 3360): META-INF/MANIFEST.MF

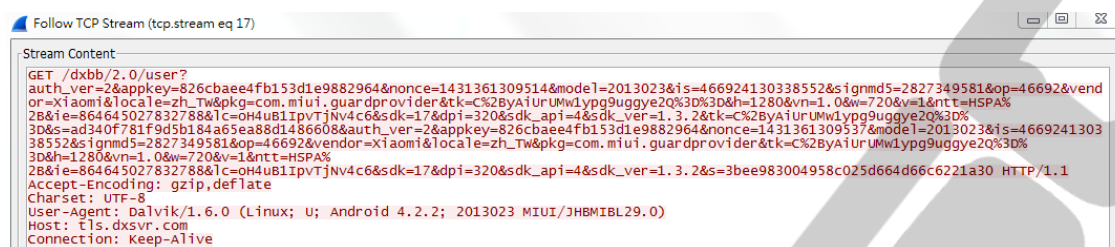
```

圖 12 com.miui.guardprovide 動作



```
05-13 16:44:09.361 I/System.out( 3360): close [socket][/0.0.0.0:41564]
05-13 16:44:09.360 I/System.out( 3360): [CDS]close[41564]
05-13 16:44:09.136 I/System.out( 3360): [CDS]rx timeout:20000
05-13 16:44:09.136 I/System.out( 3360): [socket][/10.202.84.3:41564] connected
05-13 16:44:08.268 D/Posix ( 3360): [Posix_connect Debug]Process com.miui.guardprovider :80
05-13 16:44:08.268 I/System.out( 3360): [CDS]connect[tls.dxsrv.com/180.97.33.177:80] tm:20
05-13 16:44:08.268 I/System.out( 3360): [socket][0] connection tls.dxsrv.com/180.97.33.177:80;LocalPort=41564(20000)
05-13 16:44:08.267 I/System.out( 3360): property Value:true
05-13 16:44:08.266 D/libc-netbsd( 3360): getaddrinfo: tls.dxsrv.com get result from proxy >>
```

圖 13 com.miui.guardprovide 主要連線位址



```
Follow TCP Stream (tcp.stream eq 17)
Stream Content
GET /dxbb/2.0/user?
auth_ver=2&appkey=826cbaee4fb153d1e9882964&nonce=1431361309514&model=2013023&is=466924130338552&signmd5=2827349581&op=46692&vend
or=x1aomi&locale=zh_tw&pkg=com.miui.guardprovider&tk=c%2ByAiuUmw1ypg9uggye2Q%3D&h=1280&vn=1.0&w=720&v=1&ntt=HSPA%
2B&ie=864645027832788&lc=0H4u81PvTjNv4c6&sd=17&dpi=320&sd_api=4&sdk_ver=1.3.2&rt=c%2ByAiuUmw1ypg9uggye2Q%3D%
3D&s=ad340f781f9d5b184a65ea88d1486608&auth_ver=2&appkey=826cbaee4fb153d1e9882964&nonce=1431361309537&model=2013023&is=4669241303
38552&signmd5=2827349581&op=46692&vendor=x1aomi&locale=zh_tw&pkg=com.miui.guardprovider&tk=c%2ByAiuUmw1ypg9uggye2Q%3D%
3D&h=1280&vn=1.0&w=720&v=1&ntt=HSPA%
2B&ie=864645027832788&lc=0H4u81PvTjNv4c6&sd=17&dpi=320&sd_api=4&sdk_ver=1.3.2&s=3bee983004958c025d664d66c6221a30 HTTP/1.1
Charset: UTF-8
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.2.2; 2013023 MIUI/JHBMIBL29.0)
Host: tls.dxsrv.com
Connection: Keep-Alive
```

圖 14 com.miui.guardprovide 封包

將 com.miui.guardprovide 的動作分析出來看，發現固定開機就會執行並連到網路後，傳送有關手機資訊給 MIUI 主機，在手機執行每 30 分鐘後也會傳資訊給 MIUI 主機以及接收到簡訊或電話也會執行此動作，com.miui.guardprovide 的流程如圖 15。

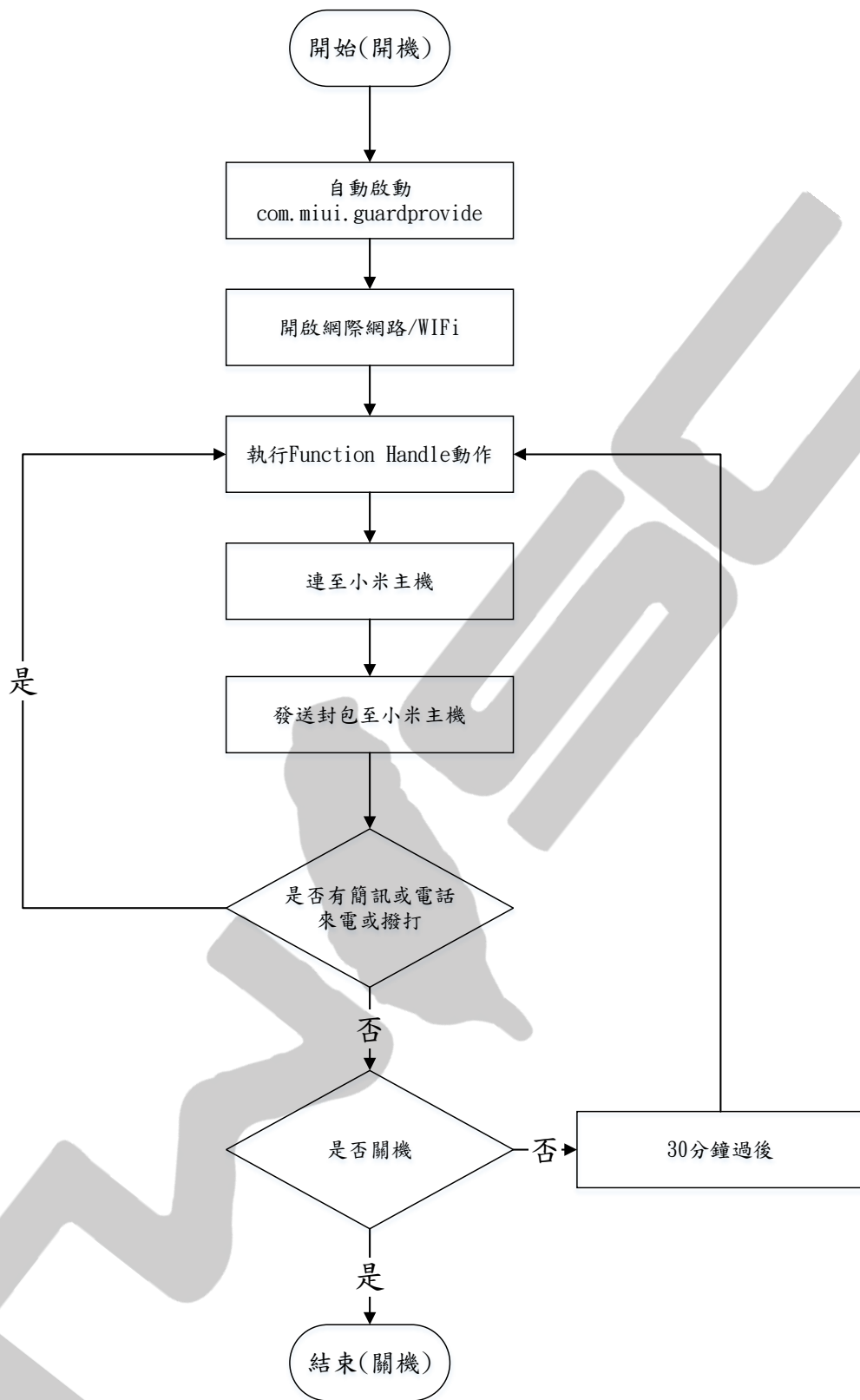


圖 15 com.miui.guardprovide 流程圖

com.mediatek.smsreg 主要啟動 android.intent.action.BOOT\_COMPLETED，意思只要開機時就啟動此 APK。

```
01-01 00:08:26.707 W/SmsReg/Receiver( 1617): Sms register is disabled by engineer mode !
01-01 00:08:26.707 W/SmsReg/Receiver( 1617): The intent is android.intent.action.BOOT_COMPLETED
```

圖 16 com.mediatek.smsreg 動作

com.android.provision 主要監控 sim 卡的狀態，也有主要的活動。

```
05-13 16:06:18.906 D/SimStateChangeReceiver( 2364): ACTION_SIM_STATE_CHANGED state : LOADED reason : null
05-13 16:06:18.793 D/SimStateChangeReceiver( 2364): ACTION_SIM_STATE_CHANGED state : IMSI reason : null
05-13 16:06:18.522 D/SimStateChangeReceiver( 2364): ACTION_SIM_STATE_CHANGED state : READY reason : null
05-13 16:04:36.023 D/SimStateChangeReceiver( 2364): ACTION_SIM_STATE_CHANGED state : LOCKED reason : PIN
05-13 16:04:35.960 D/SimStateChangeReceiver( 2364): ACTION_SIM_STATE_CHANGED state : ABSENT reason : null
05-13 16:04:35.353 D/SimStateChangeReceiver( 2364): ACTION_SIM_STATE_CHANGED state : NOT_READY reason : null
05-13 16:04:35.214 D/SimStateChangeReceiver( 2364): ACTION_SIM_STATE_CHANGED state : NOT_READY reason : null
```

圖 17 com.android.provision 動作

接下來我們觀察手機使用狀況可以發現，作業系統本身一定占用固定量的使用量 (如 com.android.systemui、com.sonyericsson.home、com.miui.home、com.htc.launcher)，而 Google 在每台手機也占用了一定的使用率主要是 Android 手機本身就有許多內建的 google 應用程式，加上每台手機都需要 google 定位與一些基本的功能，而剩下使用量為其他應用程式，其他應用程式主要就是各家手機自行的應用程式。

每台手機在執行時主要活動的 APK 以 OS、內建、Google 應用程式，在 MIUI 的部分發現內建的應用程式相對其他廠牌比起來使用的較多如圖 18。

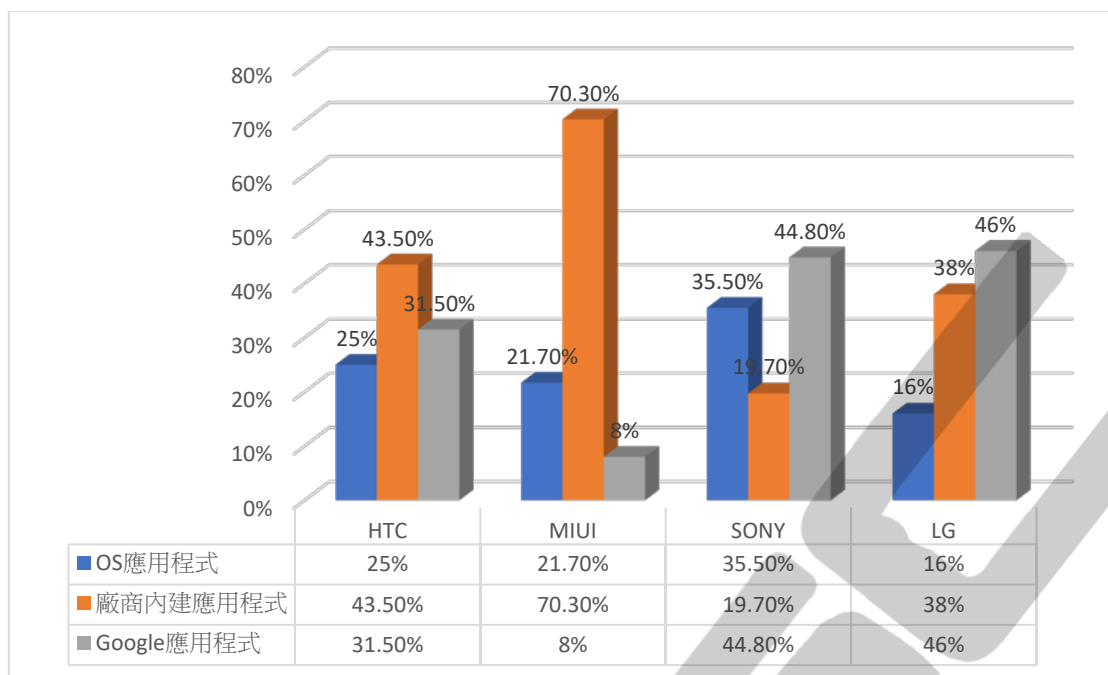


圖 18 各廠牌應用程式所占 CPU 使用量

圖 19 為手機開機連線國家，所有手機於開機時皆會與美國伺服器連線主要是 Google 的 IP 與網路校隊時間為主，有些手機會與自家廠牌的國家連線例如 Sony 開機就會與日本做連線的動作，有些手機有綁定 Facebook 因此開機時會連線到 Facebook 的主機。

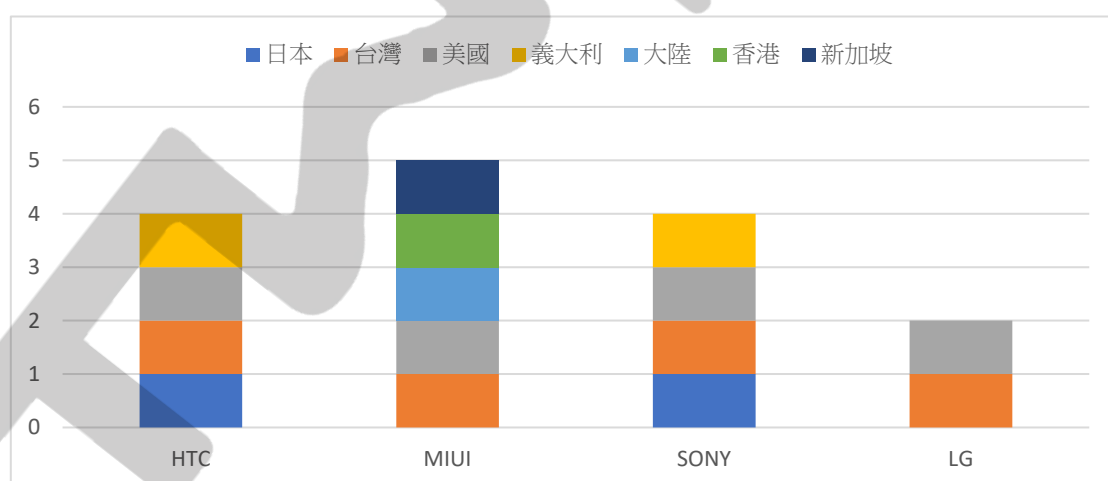


圖 19 手機開機連線國家

各家廠牌的手機主要對外連線主機的狀態，發現 MIUI 連線自家主機的 IP 非常多以及還有與百度有所連線，而在 Sony、HTC、LG 的部分連線的自家主機的 IP 與 Google

如圖 20。

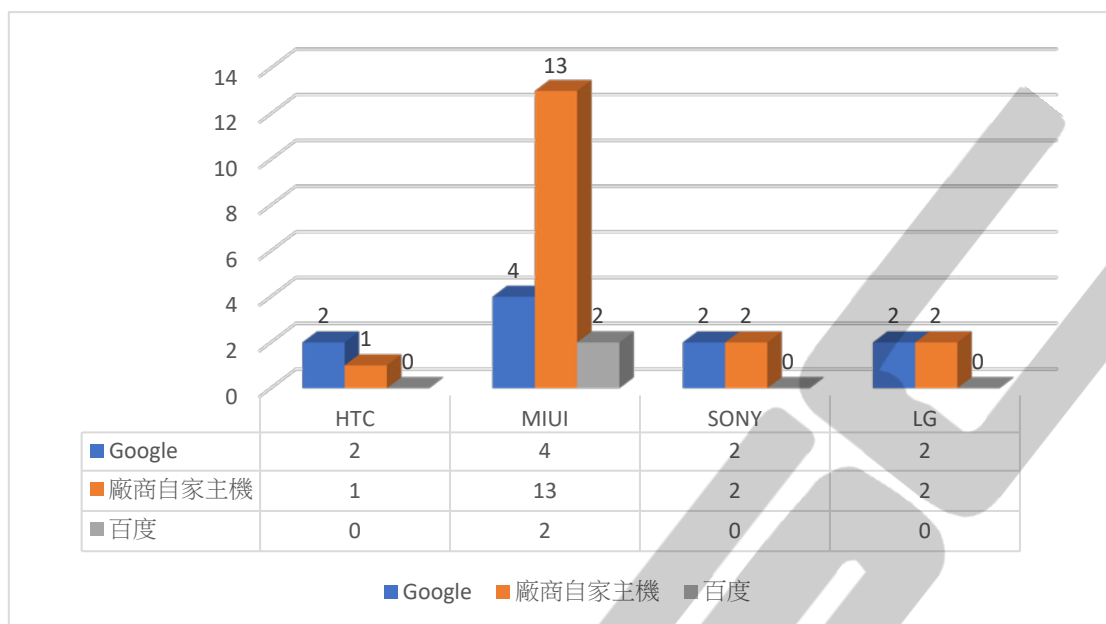


圖 20 各廠牌對外連線的主機

各家廠牌手機應用程式網路的使用狀況，在圖 21 可以看到 MIUI 以及 HTC 手機內建應用程式比起其他廠牌使用較高，因為 MIUI 雲端應用程式不斷的與 MIUI 主機連線。

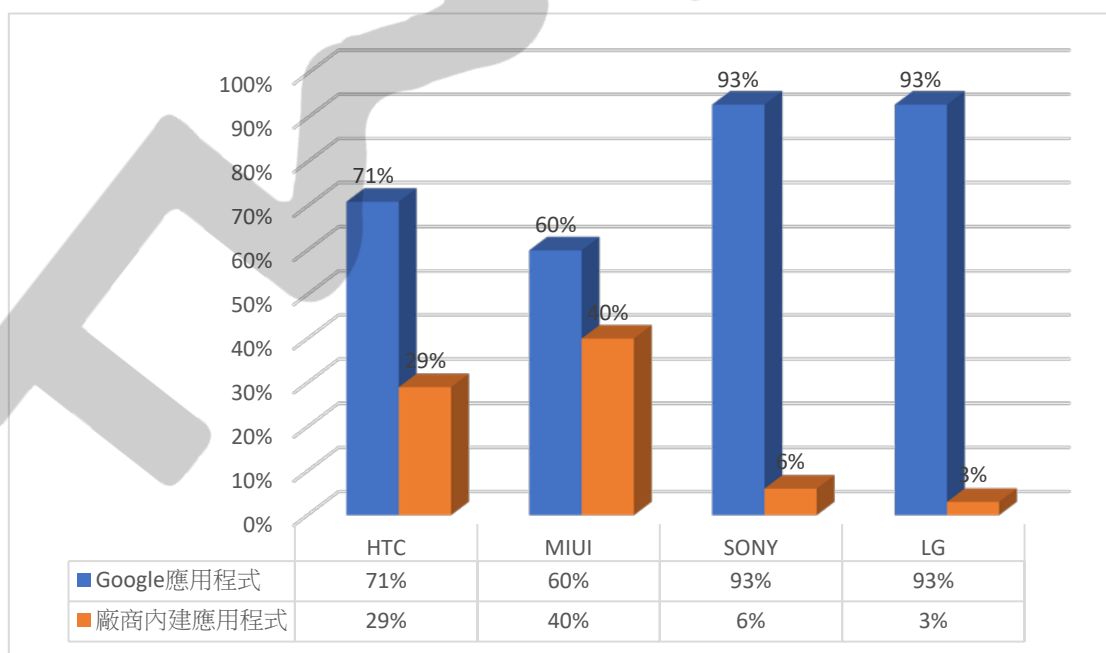


圖 21 各廠牌應用程式使用的網路狀態

在網路行為分析中每台手機的結果都不一樣，我們將手機內部應用程式每個都執行查看網路傳輸的內容，表 9 可以得知每台手機基本上都會傳輸 IMEI、手機作業系統、手機型號、MCC、MNC，其中又以 MIUI 手機傳輸的內容較多。

表 9 各廠牌手機傳輸資料內容

手機廠牌	HTC	MIUI	SONY	LG
IMEI	✓	✓	✓	✓
IMSI		✓		
GPS 定位	✓	✓	✓	
手機作業系統	✓	✓	✓	
手機型號	✓	✓	✓	✓
手機韌體版本	✓	✓	✓	✓
MCC	✓	✓	✓	✓
MNC	✓	✓	✓	✓
書籤內容		✓		
手機設定		✓		
自動下載檔案		✓		
資料加密，內容 未知		✓		

將每台手機開機時所傳輸的內容用流程圖來看，每台手機傳輸的內容都不太一樣如圖 22、圖 23、圖 24、圖 25，但一定會先與網路校對時間，傳送 GPS 定位是應用程式

要確認天氣狀況，傳送系統資訊是某些手機查看版本是否需要更新，相較之下 MIUI 手機較其他廠牌多了些傳輸內容如 LOG 檔、下載檔案等。

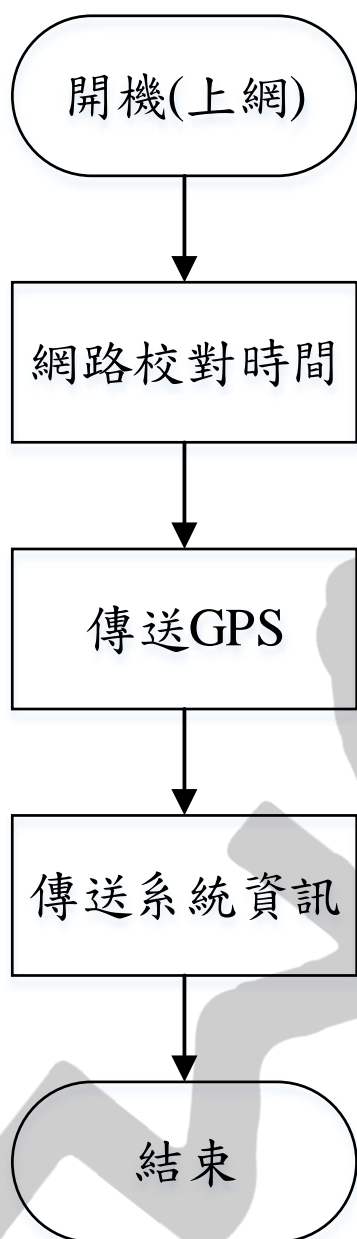


圖 22 HTC 資料傳輸流程圖

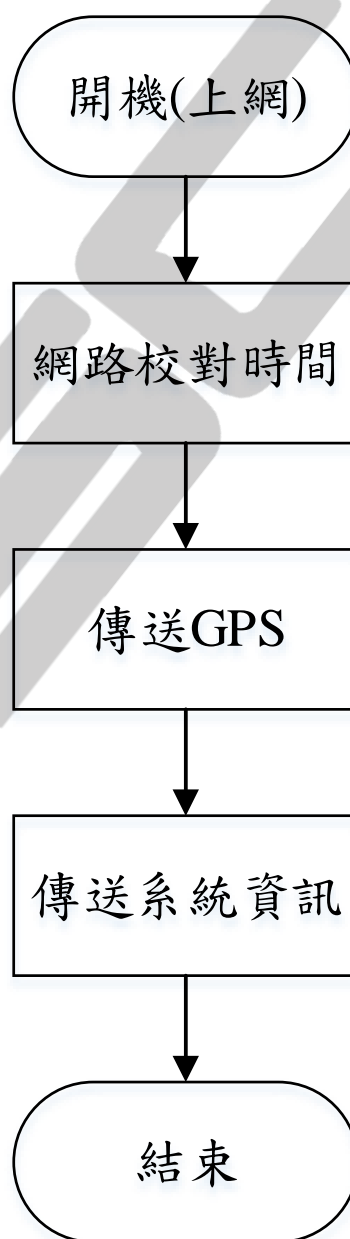


圖 23 Sony 資料傳輸流程圖

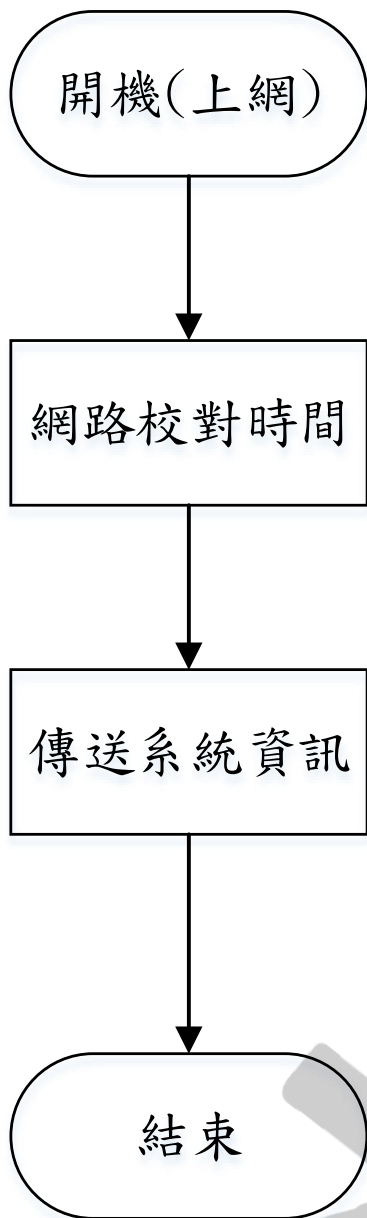


圖 24 LG 資料傳輸流程圖

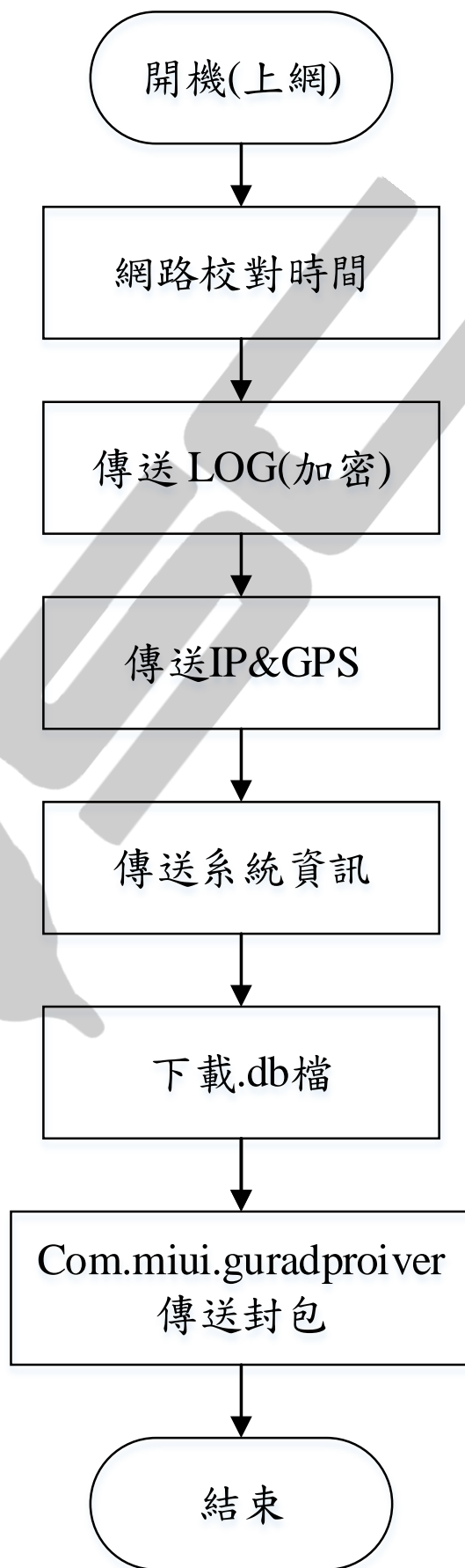


圖 25 MIUI 資料傳輸流程圖



## 綜合分析

在手機實驗中我們可以發現每台手機所檢測出來的結果都不一樣，本章節將靜態分析以及動態分析的結果彙整後與惡意程式所得到的特徵進行比對，符合惡意特徵項目越多的手機有安全疑慮的可能性更高，表 11 作為動態分析之判定手機是否為安全疑慮的基準。

表 11 動態分析之判定為有安全疑慮的基準

動態分析		有安全疑慮的行為
手機 使用狀況	CPU	符合惡意程式 CPU 使用率特徵 1、特徵 2(圖 19、圖 20)
	RAM	使用狀態超過 10000MB
	Logcat 訊息	1. 傳輸簡訊 2. 更改設定 3. 特地連線主機超過 5 個(自家主機)
網路封包	傳輸資料內容	IMEI、IMEI、GPS 定位、手機作業系統、手機型號、手機韌體版本、自行下載檔案、加密資料

## HTC

在靜態分析中 HTC 內建應用程式經由 Virustotal 檢測出結果無安全疑慮；在權限分析中 25 個有安全疑慮的權限中占 6 個為使用率最高的；在手機使用狀態中都為正常；在網路封包傳輸中有 5 個符合有安全疑慮的傳輸內容如表 12。

表 12 HTC 動靜態分析結果

分析方式		結果
靜態分析	Virustotal	0/184
	權限分析	6/25
動態分析	CPU 使用狀況	正常
	RAM 使用狀況	正常
	Log 訊息	特定連線主機 3 個
	封包分析	5/8

## MIUI

在靜態分析中 MIUI 內建應用程式經由 Virustotal 檢測出結果有三個有安全疑慮；在權限分析中 25 個有安全疑慮的權限中占 13 個為使用率最高的；在手機使用狀態中 GuardProvider.apk 應用程式 CPU 使用率與惡意程式 CPU 使用率特徵 2 相符其他皆為正常；在網路封包傳輸中有 8 個符合有安全疑慮的傳輸內容以及還傳輸手機設定等資訊如表 13。

表 13 MIUI 動靜態分析結果

分析方式		結果
靜態分析	Virustotal	3/88
	權限分析	13/25
動態分析	CPU 使用狀況	非正常
	RAM 使用狀況	正常
	Log 訊息	特定連線主機 19 個
	封包分析	8/8

## Sony

在靜態分析中 Sony 內建應用程式經由 Virustotal 檢測出結果皆為正常；在權限分析中 25 個有安全疑慮的權限中占 0 個為使用率最高的；在手機使用狀態中都為正常；在網路封包傳輸中有 5 個符合有安全疑慮的傳輸內容如表 14。

表 14 Sony 動靜態分析結果

分析方式		結果
靜態分析	Virustotal	0/215
	權限分析	0/25
動態分析	CPU 使用狀況	正常
	RAM 使用狀況	正常
	Log 訊息	特定連線主機 4 個

	封包分析	5/8
--	------	-----

## LG

在靜態分析中 LG 內建應用程式經由 Virustotal 檢測出結果皆為正常；在權限分析中 25 個有安全疑慮的權限中占 6 個為使用率最高的；在手機使用狀態中都為正常；在網路封包傳輸中有 5 個符合有安全疑慮的傳輸內容如表 15。

表 15 LG 動靜態分析結果

分析方式		結果
靜態分析	Virustotal	0/141
	權限分析	6/25
動態分析	CPU 使用狀況	正常
	RAM 使用狀況	正常
	Log 訊息	特定連線主機 4 個
	封包分析	5/8

## 結論與未來發展

現在 Android 手機不在是只有惡意程式的威脅，手機廠商有可能也會竊取使用者個人資料，目前相關惡意程式工具都使用虛擬機來偵測無法使用實體手機，因此本研究想藉由這個系統可以讓使用者只要下載並將手機連接 USB 插上電腦就可以使用，以及提出一種 Android 手機檢測方法，讓檢測環境容易並能夠檢測出手機的安全性。

在本研究中，使用了靜態分析與動態分析方法對實體手機進行檢測，藉由惡意程式的特徵以及行為，來判別真實手機是否有安全上的疑慮，下列將四個廠牌手機有安全疑慮的特徵提出：

- (1). HTC：有安全疑慮的為權限應用因有些應用程式要求過多的權限，而在特定主機連線次數為 3 個(主要為 GOOGLE、HTC)，在封包分析中主要傳輸資料有 IMEI、GPS 定位、手機作業系統、手機型號、手機韌體版本其他皆為安全。
- (2). MIUI：靜態分析 Virustotal 檢測出三個應用程式有安全疑慮；權限與其他廠牌相比要求權限最多；封包分析中傳輸資料為 IMEI、IMSI、GPS 定位、手機作業系統、手機型號、手機韌體版本、MCC、MNC、書籤內容、手機設定、自動下載檔案、資料加密，對外連線主機為 19 個(主要為百度、MIUI、GOOGLE 主機)；手機使用狀況 CPU 使用率符合惡意程式 CPU 使用的特徵。
- (3). Sony：靜態分析些為安全，權限分析要求也是四個廠牌中最少的；封包分析傳輸資料為 IMEI、GPS 定位、手機作業系統、手機型號、手機韌體，對外連線主機為 4 個(主要為 Sony、GOOGLE 主機)其他皆為安全。

(4). LG：有安全疑慮的為權限有些應用程式要求過多的權限，而在特定主機連線次數為 4 個(主要為 GOOGLE、LG 主機)，在封包分析中主要傳輸資料有 IMEI、GPS 定位、手機作業系統、手機型號、手機韌體版本其他皆為安全。

在實驗結果我們能夠發現在所有的品手機廠牌中，MIUI 手機不論在特徵碼分析、權限分析、網路行為以及手機狀況，相較與其他廠牌手機有較高的危險性，而其他廠牌也有一些安全疑慮的存在不過與 MIUI 比較起來安全率較高。

本研究並沒辦法完全證明使用這台手機是惡意的，只是提出一種概念性驗證，在網路行為方面有些資訊是有經過加密處理無法完全了解內部的內容，而本研究對於每台手機測試得時間沒有很長，有些惡意程式並需要在長時間下才拿觀察出來，本研究只能進一步讓使用者當參考用，未來研究的方向可以將手機測試的時間拉長，並且搭配有無 SIM 卡以及 WIFI、3G、4G 不同模式下去進行檢測以及檢測應用程式儲存安全性有時儲存不適當也會造成資料陷入危險。