Android 智慧行動裝置之安全管控機制之研究與開發

張佑嘉、邱奕斌、伍立鈞、王傑民、吳育松
交通大學資訊工程學系

摘要

對於攜入個人智慧行動裝置(Bring Your Own Device)於軍事管制區，目前國防部已著手導入行動裝置管理(Mobile Device Management)第三方解決方案來處理其所造成的資安隱憂。但由於管制區內的人事物有其敏感性，若僅仰賴第三方解決方案，恐形成一極大的國安漏洞。因此在本研究中我們對Android智慧行動裝置安全管控各項功能之實作可行性進行評估進而掌握其背後所需的關鍵技術，包括如裝置資訊取得、裝置控制、管控系統自體防護以及架構設計等。本研究的成果可立即作為國防部於初期導入第三方MDM解決方案的採購評量參考，而針對中長期須自行開發MDM系統的目標，本研究所發展的諸項關鍵技術也將具有極高的參考價值。

關鍵字：行動裝置管理、員工自帶設備上班、安全性、軍事管制區

System Configuration Extraction and Management for Android Devices

Yu-Jia Zhang, Yi-Pin Chiu, Li-Jiun Wu, Chieh-Min Wang, Yu-Sung Wu
Department of Computer Science,
National Chiao Tung University

Abstract

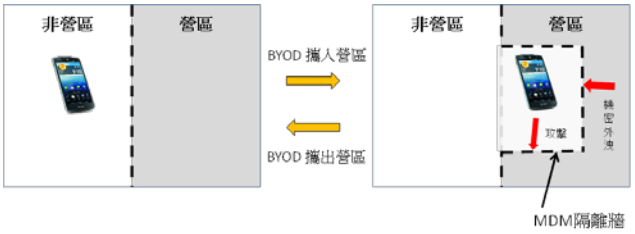
In view of the security concerns caused by Bring Your Own Device (BYOD) in military controlled zones, the department of defense is now in the process of evaluating third-party mobile device management (MDM) solutions to address the concerns. However, due to the sensitive nature of military controlled zones and the widespread usage of mobile devices across all ranks of military personnel, the trustworthiness of a third-party MDM is no doubt a security threat by itself. In this research, we developed the technologies for building an in-house MDM solution, which include device status checking, device function control, MDM agent deployment, and MDM self-protection mechanism. We also built a prototype MDM system. The evaluation results confirmed that the system is practical and effective.

Keywords: mobile device management, bring your own device, security, military controlled zone

1. 研究動機

智慧型行動裝置在我國的普及率已逾 30%，資策會更預測在 2015 年前，智慧型行動裝置的普及率將突破 50%。智慧型行動裝置已儼然成為一般人生活中不可或缺的一環，然而在軍事管制區內允許使用智慧行動裝置卻也造成諸多安全隱憂，包括如有心人士可藉由其照像、定位、以及連網能力來進行情蒐等問題、抑或是敵方人員可透過具有安全漏洞的裝置對軍事情報網路進行滲透等問題。也因此目前智慧行動裝置在軍事管制區內的使用是有著非常嚴格的限制的，而這也造成了國軍人員生活上的諸多不便。

允許人員攜帶個人智慧行動裝置進入管制區，也就是所謂的 Bring Your Own Device(BYOD)已是一個不可逆轉的趨勢，而這樣產生了智慧終端裝置管理機制 Mobile Device Management (如圖表 1)之需求。目前市面上已有諸多商用 MDM 解決方案。以初期來說，國防部可透過導入這些方案來解決眼前對於 BYOD 之安全管控的迫切需求。但另一方面，基於國防技術要能自主掌握的原則，僅依賴商業解決方案並非長久之道，特別是 BYOD 議題所涵蓋的範圍上可至三軍統帥，下可至基層士兵，若不慎被敵人利用，所造成的衝擊將會是全面性的災難，也因此我們對於 MDM 技術的掌握有其絕對的必要性，中長期而言我們也必須要有建置自主 MDM 系統的能力，而非僅仰賴商用產品來處理 BYOD 之需求。

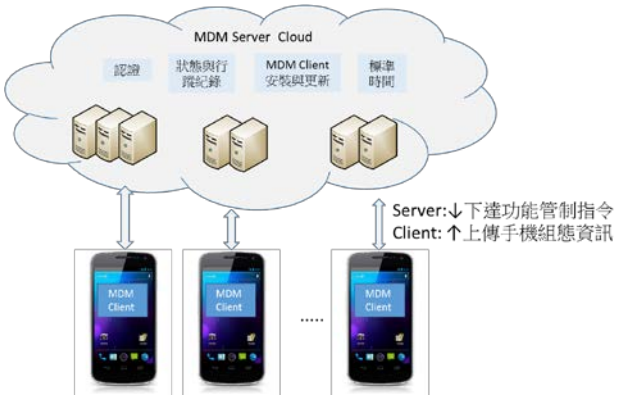


圖表 1. Mobile Device Management 示意圖

2. 系統設計

系統之架構如圖表 2所示，整個系統大致可分為 MDM 伺服器端(MDM Server)與 MDM 客戶端(MDM Client)。整體的管控與檢測流程簡述如下，首先在手機裝置上安裝我們所設計的 MDM 客戶端應用程式並進行註冊登入程序後，MDM Client 會開始對手機進行管控，藉此避免其違反管制區的資安政策。MDM client 將透過 Android 本身的管控 API，以及定期對於手機組態資訊的檢查來達到安全管控的目的。MDM client 亦會將裝置資訊定期回報給 MDM 伺服器，藉此來掌握管制區內所有智慧行動裝置之行蹤以及狀

態。

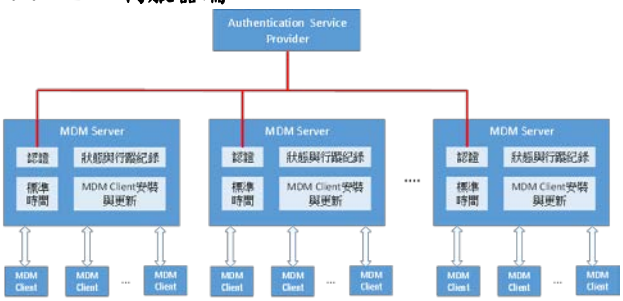


圖表 2. MDM 系統架構

2.1. MDM 客戶端應用程式

MDM Client 安裝於國軍人員的行動裝置上，會定期檢查行動裝置的組態資訊，包括應用程式組態資訊(Applist)和系統組態資訊(System) 。其中 Applist 包含行動裝置上 App 的軟體資訊(版本、版本號、更新時間等等)，而 System 的部分則是包含手機的系統資訊(包括 IMEI、WIFI、GPS、3G、Camera、藍芽、Mic 等)，MDM Client 會定期的將這些手機組態資訊上傳至 MDM Server， Server 端即可藉由這些資訊得知該裝置是否有違反軍中管制區所訂立的操作規則，若是發現手機可能會對軍事管制區造成安全上的隱憂，Server 端可以即時的下達管控手機的指令，並在 MDM Client 端做實際的裝置管控的操作。

2.2. MDM 伺服器端



圖表 3. MDM Server 架構圖

在 Server 的架構上考慮到士兵可能會在不同的營區間移動，為了達到在不同 Server 仍可對士兵行動裝置進行身分認證，我們需要一個中央 Server 來控管身分。上圖圖表 3即為 MDM Server 的系統架構圖，如圖所示在系統架構上最上層為一提供認證控管機制的中央 Server（Authentication Service Provider），行動裝置在使用本系統執行控管前都需要先通過此中央 Server 的認證才能完成註冊，而架構第二層則是各營區所配置的 MDM Server，負責監控 MDM client 的各狀態以及位置資訊，並能夠在需要的情況下對 MDM client 下達指令。在整個系統的運作上，當系統要開始管控進入營區的行動裝置時，會需要先對行動裝置進行認證的動作，此時營區的 MDM Server 會將認證的資訊導向中央 Server 做處理，當認證通過後 MDM Server 即會進一步地與 MDM client 溝通並傳輸 client 所擷取到的裝置狀態資料來做狀態監控。

2.3. 系統連線與認證機制

2.3.1. 系統連線

在實作上我們的伺服器端用 Java 寫成，因為手機端是採用 Android 系統，可直接使用 Java 的函式庫，則可以使用同一套函式庫應用在伺服器端、客戶端上，可以有較好的便利性和相容性。我們使用 Apache XML-RPC[1]來讓客戶端調用伺服器端的函式，XML-RPC 是 RPC 的一種實作方式，它藉由在 HTTP 通道中交換 XML 格式的資料來調用伺服器端的函式和回傳結果，在 Java 中最常被使用的 XML-RPC 函式庫就是 Apache XML-RPC，同樣的它也可以在 Android 中使用。

由於 XML-RPC 是使用 HTTP 通道，所以可以透過 SSL/TLS 進行加密通訊；當每次呼叫 RPC 時，XML-RPC 都會完整的建立一次 SSL 和 HTTP 連線，而預設上每一次的 RPC 都會重新產生一個被呼叫得類別，這代表每一次的 RPC 彼此是不相關的，但有時候會需要彼此相關的函式呼叫，這時 XML-RPC 也有提供 Cookie 的功能，可以保存狀態。

2.3.2. 識別碼生成

為了避免 Server 被未授權的 MDM client 進行攻擊，我們希望所有連線的 MDM client 在第一次安裝時都必須要先經過「註冊」的動作，MDM server 也僅針對註冊過後的裝置提供服務，來避免惡意攻擊，而這些註冊動作都必須在國防部的區域網路下完成，以確保僅有國軍能夠註冊使用 MDM server。而為了識別每一台不同的行動裝置，我們的註冊動作便是取得每一台裝置不同的 unique id 並記錄於 MDM Server 上，往後的連線便皆需要核對此 unique id 是否存在於 MDM Server，作為身分驗證。在 Android 系統上能夠識別不同裝置的 unique id 有以下幾種(1)手機的 IMEI 識別碼[2]、(2)網路卡或藍芽裝置的 MAC Address、(3)每一個 Android 作業系統上都存在的 ANDROID ID。第一種手機的 IMEI 識別碼存在於所有的手機上，作為電信商區別不同手機的唯一識別碼，手機出廠後便無法變更，作為安全防護使用的 unique id 十分合適，但是考量到不具備 IMEI 碼平板電腦也有可能需要列入管控，或者是具備兩個 SIM 卡模組的雙卡手機，就會有兩個 IMEI 碼，所以還需要其他方式輔助；第二種使用網路卡或藍芽裝置的 MAC Address 也是一種十分有效的方法，MAC address 是網卡或藍芽連接裝置出廠便具備的唯一識別碼，但是由於可能有些特殊 Android 裝置具備有兩張以上的網卡，或是某些 Android 系統的桌上型電腦或小筆電具備可替換網卡的特性，有以仍然有所缺陷；第三種 ANDROID ID 是 Android 系統在安裝時會自動產生的唯一識別碼，但是若是重新安裝

Android 系統，此識別碼便會改變。因此我們認為最好的方式便是同時使用這三種唯一識別碼，若三種皆符合便可以正常與 MDM Server 連線，若是有一至二項不符合便會發出警訊，要求盡快重新註冊。

2.3.3. 認證協議

在認證機制的方面我們使用了 OAuth 2.0[3]作為我們的認證協議，OAuth 2.0 能夠安全地允許使用者讓第三方應用程式存取該使用者在某一網站上所儲存的受保護資源，因此 OAuth 2.0 實際上是個認證加授權的協議，在本系統中我們主要強調的是 OAuth 2.0 提供的認證功能，而上述提到的使用者在我們的系統下對應到的是 MDM client，第三方應用對應到營區 MDM Server，而儲存受保護資源的網站則是中央控管 Server。

在 OAuth 2.0 的 spec 下定義了四種獲取 Access Token 的流程，而本系統使用的流程方式為”Resource Owner Password Credentials Grant”，此流程假定第三方應用為可信任的程式，而由於在我們的應用情境下第三方應用即為營區的 MDM Server，此 Server 也是我們自行架設的因此符合信任第三方應用的假設。在此流程下使用者可以將使用者帳密交由第三方應用直接去與 Authentication Server 做認證獲取 Access Token，在我們的實作上我們使用”OAuth for Spring Security[4]”此 framework 來架設 OAuth 2.0 Service，並且將 Authentication Server 與 Resources Server 都建置在同一中央控管 Server 下，MDM client 一開始會先將使用者的帳號密碼送給營區的 Server，由於我們採用中央控管認證的機制因此營區的 Server 會將此使用者帳密加上額外的資訊集結成 Authentication Request 送往中央 Server 做認證，當認證通過後中央 Server 會回傳一 Access Token，接著 MDM Server 即可以此 Token 對中央 Server 存取士兵的資料，完成整個認證的流程。

3. 研究方法

3.1. 獲取手機端組態資訊

3.1.1. 網路連線資訊

智慧型手機要連接網路無非是使用 3G 連線或者是 WIFI 熱點，在沒有 WIFI 熱點的情況下，裝置就會轉換成 3G 連線上網。而在對於智慧型手機攜入營區，必須要確保智慧型手機是經由受信任的網路進行聯網，意即我們要能監控手機的連網的狀態。而這部分需要關注的點有手機目前的 WIFI 連線情況以及 3G 網路連線狀況的取得。

WIFI 連線的部分我們需要掌握的是 SSID(Service Set ID)以及 BSSID(Basic Service Set ID)此兩資訊，SSID 是指所連無線網路的網路名稱，而 BSSID 則是連線的無線網路裝置的乙太網路 MAC 位址，有了這兩資訊我們即可確立行動裝置的 WIFI 連線是否是經由受信任的裝置進行連網。而 3G 的部分，我們則是可以藉由取得電信網路國別、電信公司代號以及電信公司名稱等資訊來確定 3G 連網的電信公司為軍中合作的電信業者，另外若軍中規定手機不能開放行動裝置熱點分享的功能(Hotspot)，那我們也必須隨時監控熱點功能的服務狀態。

3.1.2. 時間取得

時間確認非常的重要，如果某些功能是有時段限制的，那智慧型手機的時間跟軍方標準的時間一定要同步才能達到此規範，因為時間可以劃分成很多時區，不管使用者是有意或無意更改時區使得與軍方標準時間是不同步，這都是不允許的。

為了保持 MDM Client 和 MDM Server 上的時間一致，並且避免使用者擅自更改 Android 裝置上的系統時間，導致事件紀錄混亂，我們會要求 MDM Client 所有與 MDM Server 的訊息傳遞都以 MDM Server 上的時間為基準，並且定期強制與 MDM server 進行時間同步，若是使用者關閉聯網功能、間隔太久沒有與 MDM Server 進行同步，便會自動鎖定裝置，不讓使用者進行操作，保護機密資料的安全。

另外為了避免不能跟 MDM Server 同步的時候也可以確保時間的正確性，MDM Client 會有一個 TrueTime 的機制，這可以確保在手機上使用者修改手機時間但 MDM Client 還是有一個標準時間來判定目前 Server 的正確時間， TrueTime 機制會在手機一開始進入營區進入認證後，就在那時與 Server 同步開始設一個起始點，並以這起始點為基準自行運作一個 clock tick 來加以計算目前 Server 的時間，藉此來達到在手機不連網情況下還可以與 server 正確時間同步。

3.1.3. 座標位置取得

在 Android 中有 3 種座標位置的取得方式，他們被整合在 LocationManager[5]這個類別中，分別是 GPS、Wi-Fi 定位以及 Cell ID 定位。GPS 需求的啟動時間較長，但精準度高，另外兩種精準度較低，但大概 1 秒左右就可以抓到座標位置。在我們的應用當中，對於資訊安全性的要求很高，並不希望座標位置被第三方得知，由於使用 WiFi 定位和 Cell ID 定位都需要經過 Google 伺服器，所以並不適用，則我們簡單的使用 GPS 作為座標的取得方式。

但這又產生了新的問題-「GPS 座標是否能被偽造？」在 Android 文件中，有所謂的模擬位置(Mock Location)選項，這是提供給開發者進行 GPS 相關應用程式開發所使用的選項，它讓開發者可以透過新增一個 GPS Provider 來偽裝 GPS 位置，但前提是要擁有 Mock Location 權限。如圖表 4，模擬位置可以透過加入 Test Provider 來達成，但一旦加入 Test Provider，原本實體的 GPS Provider 就會被遮蔽，這時 Android 系統中的所有程式都會抓到模擬出來的位置，經過實際測試，目前還沒有辦法讓單一程式得到真正的 GPS 位置。

```
LocationManager lm = (LocationManager) getSystemService(Context.
LOCATION_SERVICE);

lm.addTestProvider(LocationManager.GPS_PROVIDER, false, false,
        false, false, true, true, true, 0, 5);
lm.setTestProviderEnabled(LocationManager.GPS_PROVIDER, true);
lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, this);
```

圖表 4 加入 TestProvider 遮蔽實體位置

基於系統安全性，我們需要避免其他程式使用模擬位置，我們可以檢查開發者選項中的模擬位置選項，如果有被打開，則我們會要求使用者把模擬位置關閉，但這會讓所有的程式都無法模擬位置，包括我們自己的應用程式；若我們希望只有我們的應用程式能夠使用模擬位置，我們則可以透過偵測應用程式所宣告的 Permission 來達成，若有其他應用程式要求 android.permission.ACCESS_MOCK_LOCATION，則我們會強制將此程式放置背景不讓使用者正常執行該程式，此方法在3.2.3章節會詳細介紹。

表格 1. 管控功能對照模式表

	模式一	模式二	模式三
GPS			✓
相機	✓		
BlueTooth		✓	
3G		✓	
錄音			✓
熱點分享			✓
新增硬體		✓	

3.2. 手機端組態資訊管控

在手機端組態資訊管控上，我們採取了三種不同的方法以達到裝置功能的管控，此三種方法分別為：

1. 模式一：將 MDM Client 註冊成 Device Administrator[6]，並藉由 Android 系統的 Device Administration API完全關閉該系統功能。
2. 模式二：MDM Client 註冊一 Android BroadcastReceiver[7]監聽器監控被管控功能的狀態，當功能狀態發生改變後系統會發出一callback通知MDM Client做處理，我們即可在使用者欲開啟系統功能時立即做功能關閉的動作。
3. 模式三：MDM Client利用註冊Service[8]服務執行周期性的掃描機制，若發現被管控的功能遭到開啟則執行對應關閉的動作，另外我們也能夠利用此模式來強制關閉擁有非法權限的應用程式。

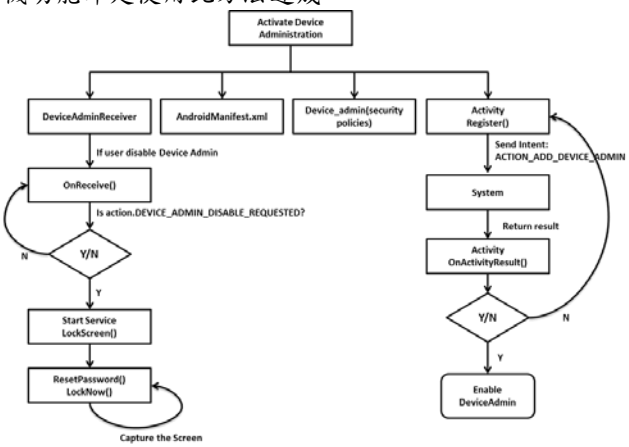
以上這三種管控模式在管控能力上為模式一最有效，能夠直接禁止功能的使用，MDM Client 只要進行一次關閉的動作，接著是模式二當每次功能狀態被改變後會觸發 MDM Client 做即時關閉，而管控上最有可能浪費系統資源則是模式三的周期性掃描機制，即使沒執行任何管控功能的操作此周期性掃描機制仍須執行因此造成不必要的資源消耗。

然而在本研究中欲管控的功能並非是使用單一模式即可完成功能的管控，模式一的管控必須是 Device Administration API 本身有提供的功能，而模式二也必須是功能本身有提供 Event-Driven 的 Callback 供我們使用。在經過我們的研究後得到了下表表格 1 的模式對應關係，以下章節詳細介紹各模式的細節以及能支援的功能管控。

3.2.1. 模式一：Device Administrator 機制

模式一是透過將 MDM Client 註冊成為 Device Administrator 進而讓 MDM Client 應用程式可以呼叫 Device Administration API 來達到管控功能的目的，能夠達到的管控功能包含了清除手機資料、立即鎖定螢幕以及關閉相機功能等操作。

實作上操作的流程圖如下圖圖表 5所示，首先需要修改 AndroidManifest.xml 此設置檔宣告 BIND_DEVICE_ADMIN 權限，並且將欲使用的管控功能宣告在 res/xml/device_admin 裡，將這些權限宣告完後再經由使用者的手動確認即可將 MDM 應用程式註冊成為 Device Administrator，使得 MDM Client 可以呼叫 Device Administration API。在實作上管控相機功能即是使用此方法達成。



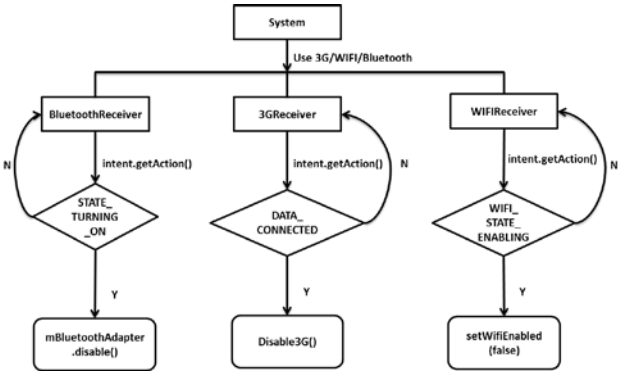
圖表 5. 模式一流程圖

3.2.2. 模式二：BroadcastReceiver 機制

模式二是 MDM Client 註冊一監聽器 Android BroadcastReceiver，當發覺到有應用程式使用到某系統功能或是使用者想開啟這些功能時，系統會以 event-driven 的方式發出一 callback 告知 MDM Client，MDM Client 即可在系統功能尚未開啟時立刻執行關閉該功能的指令，藉此對此裝置做管控阻止功能的開啟，模式二流程圖如圖表 6。

在本研究中利用模式二達到裝置控管的功能有 BlueTooth、3G、WiFi 以及新增硬體等功能，舉例來說，若是要對 BlueTooth 功能做控管，我們即可註冊一 BluetoothReceiver 監聽藍芽功能的狀態改變，若是使用者欲開啟藍芽功能，會觸發此 BluetoothReceiver 並傳入 STATE_TURNING_ON 事件，MDM Client 即可立即做關閉 Bluetooth 的動作，便能阻擋此次開啟

藍芽的操作，以達到裝置管控的功能。

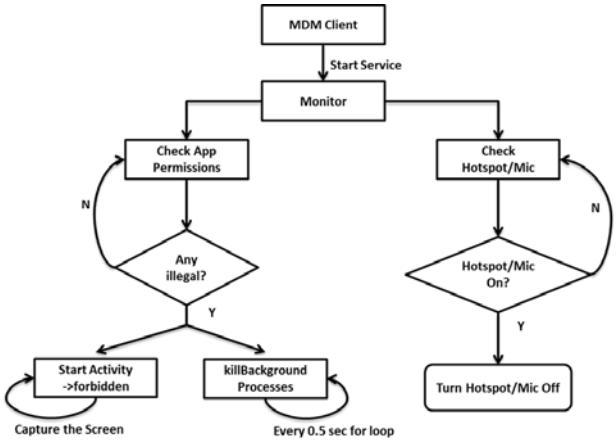


圖表 6. 模式三流程圖

3.2.3. 模式三：周期性掃描機制

模式三是透過 Service 周期性地檢查以及關閉裝置的功能，流程圖如圖表 7所示，其中應用上又分為兩種使用方式，第一種管控機制是我們可以對其他應用程式所使用的 permissions 進行檢查，如果偵測到有應用程式使用到了被管控的功能，我們即可強佔裝置的頁面不讓使用者使用該應用程式，並且執行 killBackgroundProcesses()將該應用程式強制關閉，藉此達到裝置管控的效果，而在實際應用上我們即是利用此種方式來管控 GPS 的功能，利用這種關閉其他應用程式的方式，即可使得只有 MDM Client 本身能夠獲取使用者真實的位置資訊，而其他宣告有獲取位置權限的應用程式則是無法執行。

而另一種管控方式則是以周期性地檢查系統功能是否開啟，實作上管控的功能有錄音功能以及熱點分享，MDM Client 會周期性地查詢該系統功能是否有被開啟，若為開啟狀態的話則立刻執行關閉的動作。另外在經過實驗的測量後，發現將此周期檢查的頻率設為 0.5 秒即可成功達到應有的功能管控效果。



圖表 7. 模式三流程圖

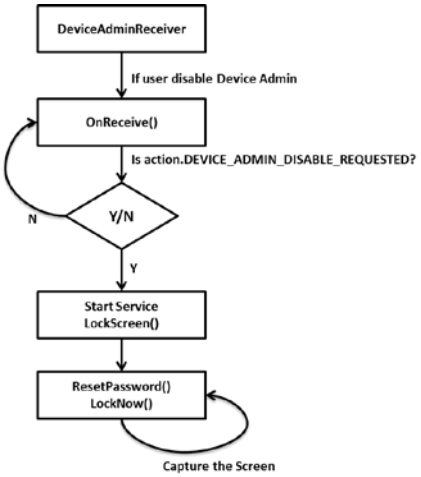
3.3. MDM Client Agent 系統防護機制

3.3.1. 手機 root 檢查

由於 Android 平台較為開放，使用者可能會透過 rooted 裝置來取得最高使用者權限，並藉此可以輕易修改系統設定，透過最高使用者權限凍結 MDM client 甚是開啟被 MDM Client 所禁止的功能，因此如同其他市面上已存在的 BYOD 方案甚至是 Samsumg KNOX[9]，我們也將不允許使用者將手機進行 root 取得最高使用者權限。MDM Client 會在第一次安裝時偵測手機是否被 root，不讓已 root 的裝置通過檢測，不允許經過 root 的裝置攜入軍營。若是使用者在攜入軍營後欲進行 root 的動作，手機勢必要進入 Recovery mode 模式，雖然我們無法藉由 MDM Client Agent 來防止使用者進入 Recovery mode，但目前 HTC 有對 Recovery mode 進行上鎖的機制[10]，因此此問題目前可行的解決方法是對以後軍方合作的手機廠商要求如同 HTC 類似的上鎖機制，若有使用者想對 Recovery mode 進行解鎖則手機廠商會自動回報給軍方人員知道，再決定是否允許此解鎖要求。

3.3.2. 防止 MDM Client Agent 被撤銷 Device Admin

由於 MDM Client Agent 在註冊 Device Administrator 時有註冊一 DeviceAdminReceiver，此 Receiver 可以監控 MDM Client Agent 目前是否有被撤銷的動作，一但偵測到撤銷事件的發生，MDM Client Agent 會立即啟動 LockScreen 機制，進行更改解鎖密碼並立即對手機進行上鎖的動作，流程如圖表 8所示，而此解鎖密碼只有軍中技術人員知道，因此使用者不能夠對手機進行解鎖的動作也無法再正常使用此手機。



圖表 8. 防止 MDM Client 被撤銷

3.3.3. 防止 MDM Client Agent 被中止執行或被反安裝

MDM Client Agent 所執行的管控模式皆會在 Service 下進行，並且此 Service 是一個 Foreground Service，此一方法類似在背景撥放音樂的音樂程式，是無法藉由 killBackground processes 來中止執行的，Foreground Service 是必須經由使用者點擊畫面來進行強制停止才能終止此 Foreground Service 執行，不過又由於我們 MDM Client Agent 已註冊成為 Device Administrator，屬於 Device Administrator 應用程式底下的 activity、service 等等的 components 皆無法被使用者藉由點擊畫面來進行強制停止的操作，因此不會

有 MDM Client Agent 被終止執行的情況。

而 MDM Client Agent 被反安裝的狀況也不會成立，因為凡是註冊成 Device Administrator 的應用程式在使用者尚未對它進行撤銷 Device Administrator 權限之前皆無法執行反安裝的動作，而由於先前提到我們會徹底防止 MDM Client Agent 被撤銷 Device Administrator(3.3.2章節介紹)，因此不需擔心 MDM Client Agent 被反安裝。

但若使用者對行動裝置進行恢復原廠設定(FACTORY RESET)的操作，在此情況下 MDM Client 還是有被移除的危機，這部分需仰賴以後軍方合作的手機廠商配合，將此功能上鎖或是取消，才能完全阻絕 MDM Client Agent 被反安裝可能性。

3.4. MDM Client Agent 安裝及更新

在 Android 系統中的應用程式分為兩大類，一類是由手機開發商在手機出廠時就預載的 system app，另外一類是使用者自行安裝的 user app，兩者所安裝的目錄、能取得的權限並不相同，system app 安裝於 Android 系統中的/system/app 中，而 user app 則安裝於/data/app 資料夾下，system app 能夠取得比 user app 更高的權限，在 Android 官方的 permission list[11]中，標有「Not for use by third-party applications」說明文字的權限，皆是僅有 system app 才能夠取得的權限。而若要達成不經由使用者的同意，在背景強制自動安裝、更新、移除應用程式，應用程式必須取得「android.permission.INSTALL_PACKAGES」此僅有 system app 才有辦法取得的權限，而 Android 的內建市集 Google Play 便是透過這個權限達到自動更新與安裝。但是要開發 system app 必須取得原始手機開發商的憑證才能夠安裝在手機上，因此除非是手機開發公司 ex: HTC、Sony、三星…等系統開發商的手機預載 App，否則是不可能達成強制安裝、更新、移除其他 App 的功能，這也是 Android 系統中防止惡意程式在不經過使用者同意的情況下安裝有危害的木馬程式，或是掉包 App 等攻擊手段的防護機制。

雖然無法強制使用者安裝、更新 MDM client，但是我們可以在 MDM client 加入版本監控的功能，定期與伺服器同步，檢查 MDM client 是否有新版本，若是使用者的 MDM client 為具有漏洞的舊版本，則要求使用者進行更新，若是使用者在限制時間內沒有進行更新，可立刻使用 Device Administration API 所提供的 remote lock 的功能，對裝置進行鎖定並更改鎖定密碼，讓使用者無法再次操作手機，並通報管理人員，達到強制更新的目的。

4. 實驗結果與分析

4.1. 實驗環境建置與測試裝置

在開發 MDM Client 的軟體方面，作業系統我們採用 Windows 7，在撰寫程式碼以及編譯的工具採用 Eclipse 這套開放原始碼的軟體，另外搭配 Java Delopment Kit(JDK) 、 Android Development Tools(ADT) 、 Android Software Development Kit(Android SDK)。

在測試裝置方面，我們採用 Samsumg Galaxy Nexus i9250 安裝 Android 系統版本 4.3、New HTC One m7 安裝 Android 系統版本 4.4.2。

4.2. 功能性驗證

在此小節我們針對所要管控的各項功能進行檢測，分別對相機、GPS、藍芽、3G、錄音、Hotspot 安裝三個不同的程式，這三支程式的採用樣本是從 Google Play 上下載有使用該系統權限的 app 並且是在 Google Play 下載率排行前三名或我們自己所開發的 app，而該 app 會企圖突破 MDM Client 的管控模式，藉由這三支程式，我們可以驗證 MDM Client 的系統管控是否能有效阻擋 app 成功使用到我們所管控的項目。

在相機功能驗證，我們採取3.2.1模式一管控，程式 1 是 Facebook，程式 2 是 Line Camera，程式 3 是美圖秀秀，操作方式是在三個程式中試圖成功拍照並能將照片存檔在手機裝置上，每支程式嘗試操作 10 次，皆無法成功拍照並存檔在手機裝置上。

在 GPS 功能驗證，我們採取3.2.3模式三情境一管控，程式 1 是 GPS 定位，程式 2 是 google map，程式 3 是我們所開發的 app，該 app 會在 GPS 開啟的狀況下，每秒向一個伺服器上傳目前 GPS 座標位置，操作方式是每當目標程式被使用者開啟時，是否能成功定位到該手機目前座標位置，每支程式各操作 10 次，皆無法成功定位。

在藍芽功能驗證，我們採取3.2.2模式二管控，程式 1 是 Bluetooth File Transfer，程式 2 是 Bluetooth Auto Connect，程式 3 是我們所開發的 app，該 app 會每秒自動開啟藍芽功能並自動配對到相鄰的指定手機裝置，操作方式是在三個程式中被使用者開啟時，是否能成功配對到相鄰的指定手機裝置，每支程式各操作 10 次，皆無法成功進行藍芽配對。

在 3G 功能驗證，我們採取3.2.2模式二管控，程式 1 是 Facebook，程式 2 是 Dropbox，程式 3 是我們所開發的 app，該 app 會每秒自動開啟 3G 功能並每秒向一個伺服器上傳檔案，操作方式是在三個程式中被使用者開啟時，是否能成功上傳檔案到伺服器，每一支程式各操作 10 次，皆無法成功進行上傳檔案。

在錄音功能驗證，我們採取3.2.3模式三情境二管控，程式 1 是超級錄音器，程式 2 是簡便錄音機，程式 3 是 Auto Recorder，操作方式是在三個程式中被使用者開啟時，使用者發出聲音，是否能成功錄音到任何聲響，每支程式各操作 10 次，皆無法成功錄到聲音。

在 Hotspot 功能驗證，我們採取3.2.3模式三情境二管控，程式 1 是便攜式 Wi-Fi 熱點，程式 2 是 Hotspot Toggle，程式 3 是安卓熱點，操作方式是在三個程式中被使用者開啟時，而程式主動觸發開啟 Hotspot，是否能成功開啟 Hotspot 功能給另外一支手機裝置連

網，每支程式各操作 10 次，另外一支手機裝置皆無法成功連上 Hotspot。

每項功能驗證阻擋率對照表如下表格 2。

表格 2. 功能驗證對照表

測試項目	程式 1	程式 2	程式 3	阻擋率
相機	V	V	V	100%
GPS	V	V	V	100%
藍芽	V	V	V	100%
Wi-Fi	V	V	V	100%
3G	V	V	V	100%
錄音	V	V	V	100%
Hotspot	V	V	V	100%

4.3. 模式三參數調整

4.3.1. 模式三的週期探討

在此小節我們針對3.2.3模式三週期性掃描的頻率探討，究竟週期要設定幾秒才能成功阻擋管控的系統功能不被執行，如以下的表格 3呈現我們會將模式三的週期參數做 5s、4s、3s、2s、1s、0.5s 調整來驗證週期參數要設定多少才是合理且適當的值，首先，我們將週期參數固定後，開始做驗證，試圖管控有使用非法系統功能的app，若有使用非法系統功能的app執行時，我們會記錄該 app 放置背景並強制停止的秒數，再經由週期參數調整後，找到週期參數設定成0.5s，能完全阻隔有使用非法系統功能的 app 成功執行。

並對每一個週期參數做占用 CPU 百分比的統計，統計的資料我們是依據當時模式三所佔用 CPU 百分比且每間隔 5s 統計一次，統計出來的結果如下表格 4，當我們模式三週期參數調整到 0.5s 時，平均對系統 CPU 的占用率僅 1.3%，說明了 MDM Client 並不會對整支手機裝置造成太大負擔，導致安裝 MDM Client 就不能正常運作手機裝置。

表格 3. 模式三間隔與停止惡意程式所需秒數

模式三間隔	停止惡意程式所需平均秒數
5s	7.5
4s	5.42
3s	3.43
2s	1.47
1s	0.25
0.5s	0

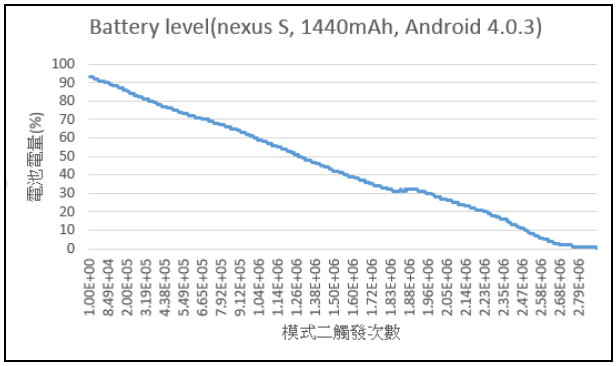
表格 4. 模式三所佔用 CPU 百分比

模式三間隔	佔用 CPU 百分比
5s	0.13%
4s	0.21%
3s	0.28%
2s	0.43%
1s	0.62%
0.5s	1.3%

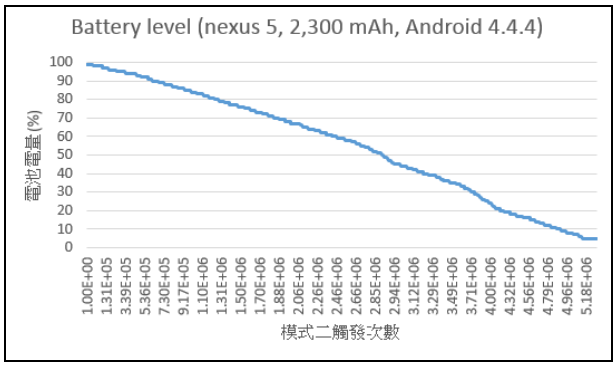
4.3.2. 模式三的耗電量

再來我們要探討模式三對於手機耗電量的影響，由於模式三是採用週期性掃描，所以必定會增加手機待機時的耗電量，也可以預想模式三會式 MDM Client 最耗電的一個管控項目。我們在這實驗中採用壓力測試，我們直接讓手機平凡的執行模式三中的函式，並去紀錄模式三函式被呼叫的次數與剩餘電力。

在這裡跟前兩個實驗不同，我們使用了兩種 CPU 資源和電量差異懸殊的 Android 裝置，分別是 Samsung Nexus S (1440mAh, A8-1core)和 LG Nexus 5(2300mAh, Krait-4core)，這是為了實驗 MDM 在不同的手機設備上，是否會過度耗費電量。下圖表 9 和圖表 10分別是 Nexus S 和 Nexus 5 的耗電量，由於 Nexus S 本身儲存電量比較小，所以它只能支撐到 280 萬次的模式三觸發，Nexus 5 則可以持續到 520 萬次的觸發。由於我們是 0.5s 才觸發一次，所以即使是電量較小的 Nexus S，都可以支撐到 140 萬秒(不計算待機電量的情況)，這代表模式三的耗電量對於整個 Android 手機的耗電量來說幾乎是微乎其微的，MDM Client 並不會減少手機本身的待機時間。



圖表 9. Nexus S 觸發次數與剩餘電量



圖表 10. Nexus 5 觸發次數與剩餘電量

4.4. MDM Client Agent 防護

在此小節我們將驗證 MDM Client 的 Device Administrator 權限能否恣意被使用者所關閉，由於 Device Administrator 權限是能被使用者在手機設定自行撤銷的(如圖表 11)，所以我們會在 MDM Client 接收到要被撤銷的瞬間就對手機做出更改密碼跟鎖定螢幕的管控，使得使用者無法恣意的關閉 Device Administrator 權限，以下實驗分別找了五人，幫我們

實測能否透過手機設定將 MDM Client 的 Device Administrator 權限撤銷掉，每人操作十次，五人共計五十次，皆無法將 MDM Client 的 Device Administrator 權限撤銷。

圖表 11. 撤銷 Device Administrator

另外再探討 MDM Client 是否能被移除，實驗統計表如表格 5，我們一樣用三個程式來實測是否能被移除，這三個程式採用樣本是從 Google Play 上下載有使用該系統的 app 並且是在 Google Play 下載率排行前三名，程式一是輕鬆卸載軟體，程式二是完美卸載器，程式三是 App 管理大師，皆無法成功地將 MDM Client 移除。

表格 5. 移除 MDM Client

測試項目	輕鬆卸載	完美 卸載器	App 管理大師
移除	X	X	X

5. 結論

智慧型手機在現代已經是生活上不可或缺的裝置，往後在軍中也要使用手機也是必然的趨勢，但軍中有太多的軍事機密是不能洩漏的，而智慧型手機的功能越來越完善，帶進軍中造成機密洩漏不可不防。

本論文秉持這樣的原則下開發了一套 MDM 系統，藉由 MDM Server 和 MDM Client 互相的溝通配合來達到完善的手機功能管控，管控模式有三種，藉由三種模式的交替使用我們能成功地管控住可能會造成機密洩漏的系統功能，而且對於 MDM Client Agent 系統防護機制我們也深入了探討許多議題並且提供解決之道。

目前尚未有任何一篇論文提及管控手機系統功能的技術性，且國防部目前僅依賴商業解決方案並非長久之道，基於國防技術要能自主掌握的原則，我們也會將原始碼公開給國軍單位，幫助國軍早一步讓智慧型手機引進軍中。

6. 未來發展

目前 MDM 雖然已經能夠良好的運作，但模式三只能算是一個折衷的方案，雖然我們找到了能良好運作的間隔時間並且也證明它對耗電和 CPU 佔用率非常低，但我們仍希望能夠提供一個完美的方案，也就是在 App 完成開啟前就進行防堵，這需要對 Android 核心及其 Framework 進行一定的了解和修改。

若在提供自製 Android OS 的前提下，許多只依賴 Android SDK 不能達成的功能將可以實現，例如直接防止黑名單 App 開啟、常態性的對整支手機做加密(使用者無法解開)、強制刪除系統 App 等等。此種作法不僅能夠提供更完全的防護，也可以有效的減少耗電量和 CPU 使用率。

7. 誌謝

感謝所有協助103年度國防科技學術研討會的相關人員。

8. 參考文獻

[1] S. A. Phillip Merrick, Joseph Lapp. *XML-RPC*. Available: https://www.google.com/patents/US7028312

[2] Wiki. *IMEI*. Available: http://zh.wikipedia.org/wiki/IMEI

[3] Eran Hammer-Lahav. *OAuth 2.0*. Available: http://oauth.net/2/

[4] The Spring Team. *Spring Security OAuth*. Available: http://projects.spring.io/spring-security-oauth/

[5] Android Developers. *LocationManager*. Available: http://developer.android.com/reference/android/location/LocationManager.html

[6] Android Developers. *Device Administration*. Available: http://developer.android.com/guide/topics/admin/device-admin.html

[7] Android Developers. *BroadcastReceiver*. Available: http://developer.android.com/reference/android/content/BroadcastReceiver.html

[8] Android Developers. *Service*. Available: http://developer.android.com/guide/components/services.html

[9] Samsung. *Samsung KNOX*. Available: http://www.samsung.com/global/business/mobile/platform/mobile-platform/knox

[10] HTC Corporation. *HTC unlock*. Available: http://www.htcdev.com/

[11] Android Developers. *Permissions*. Available: http://developer.android.com/reference/android/Manifest.permission.html