Android 應用程式安全性動態分析

A Study on Dynamic Security Analysis for Android Applications

郭忠義、張天瑋
國立臺北科技大學資訊工程系
Jong Yih Kuo and Wei-Tien Zhang
Department of Computer Science and Information Engineering,
National Taipei University of Technology
Email: jykuo@ntut.edu.tw

摘　要

因近年智慧型行動裝置的普及，行動應用服務迅速的發展，使得智慧型行動裝置與生活越來越密切相關，同時也吸引了駭客的注意，以使用者的隱私資料為目標來謀取利益。本研究利用動態分析系統偵測應用程式在執行期間的行為，並依據經濟部工業局所提出的行動應用 App 基本資安規範相關檢測項目與基準進行分析，並輸出成報表提供給使用者觀看，幫助使用者判斷其應用程式的安全性。

關鍵詞：Android 應用程式測試、動態分析
Keywords: Android Application Testing, Dynamic Analysis

1 緒論

隨著智慧型行動裝置的普及，行動裝置儼然成為人們最長時間使用的電子產品，大大改變人類生活模式。各家開發廠商開始在行動應用程式市場上展開激烈的競爭，其中又以 Android 平台市佔率最高。

由於行動應用程式市場的蓬勃發展，應用程式的種類與數量日益劇增，各式各樣的應用程式漸漸地與人們的生活息息相關。許多人開始透過應用程式購買商品、瀏覽網站與寄送郵件，這些應用程式提供各式各樣的服務以及良好的使用者體驗，但在這當中所包含的各種使用者的隱私資料也吸引了駭客的注意，並對其製作惡意程式，造成使用者的資料外洩或是被用來謀取利益，對使用者造成極大的損失。

為了對應這類問題，經濟部工業局委託財團法人資訊工業策進會，完成制訂「行動應用 App 基本資安規範」[1]，供業界開發行動應用程式自主遵循參考。

本論文將針對「行動應用 App 基本資安規範」所訂定的檢測項目進行研究，結合爬蟲技術與自動化測試工具，開發一個自動化動態 Android 應用程式資安檢測平台，藉由這平台來幫助使用者能夠更清楚的知到自己所使用的應用程式背後的行為。

本文的組織如下，第二節將介紹相關背景與研究，第三節介紹 Android 應用程式資動態資安分析系統的架構以及分析方法，第四節分析各實驗之結果，第五章則為結論。

2 相關背景與研究

2.1 行動應用 App 基本資安規範

為了預防開發者在開發行動裝置應用時，因缺乏相關資安意識，可能造成使用者資料外洩或財物損失之風險。經濟部工業局參考相關國際資訊安全規範，如 OWASP Mobile Top 10[2]、NIST(National Institute of Standards and Technology)的 Special Publication 800-163 Vetting the Security of Mobile Applications[3]等制定「行動應用 App 基本資安規範」，供業界開發行動應用。

該規範分成行動應用程式發布安全、敏感性資料保護、付費資源控管安全、行動應用程式使用者身分認證、授權與連線管理安全、行動應用程式碼安全等五個層面提出資訊安全技術要求，以下分別說明。

2.1.1 行動應用程式發布安全

主要適用於發布行動應用程式之相關資訊安全技術要求，目的在於確保行動應用程式應於可信任來源之行動應用程式商店發布與更新，並提供問題回報之管道。項目包括：行動應用程式發布、行動應用程式更新、行動應用程式安全性問題回報。

2.1.2 敏感性資料保護

主要適用於敏感性資料與個人資料保護之相關資訊安全技術要求，敏感性資料指行動應用程式建立或

儲存於行動裝置及其附屬儲存媒介之資訊，而該資訊之洩漏有對使用者造成損害之虞，包括但不限於個人資料、通行碼、即時通訊訊息、筆記、備忘錄、通訊錄、地理位置、行事曆、通話紀錄及簡訊等等。項目包括：敏感性資料蒐集、敏感性資料利用、敏感性資料儲存、敏感性資料傳輸、敏感性資料分享、敏感性資料刪除。

2.1.3 付費資源控管安全

主要適用於付費資源控管之相關資訊安全技術要求，目的在於應用程式對於付費資源的操作是否有經由使用者同意，項目包括：付費資源使用、付費資源控管。

2.1.4 身分認證、授權與連線管理安全

主要適用於行動應用程式身分認證、授權與連線管理之相關資訊安全技術要求，項目包括：使用者身分認證與授權、連線管理機制等。

2.1.5 行動應用程式碼安全

主要適用於行動應用程式開發之相關資訊安全技術要求，項目包括：防範惡意程式碼與避免資訊安全漏洞、行動應用程式完整性、函式庫引用安全與使用者輸入驗證等。

2.2 Android 安全性分析

Android 安全性分析一般分為動態分析與靜態分析兩個部分。動態分析是指透過實際執行應用程式，並針對 Android 設備進行監控，記錄執行時的網路傳輸、資料的流動、系統呼叫、設備的使用紀錄訊息等等，最後加以分析。大部分的情況下，動態分析是運行在獨立的沙盒環境下，進行動態分析的 Android 設備需要對於系統核心進行訂製與改寫，包含改寫安全機制、在系統核心加入監控器等等，如 Stowaway[4]、Quire[5]。

靜態分析則是透過對 Android 的應用程式進行反編譯，將應用程式的原始碼轉譯出來，透過分析原始碼的方式來進行。相較於動態分析，靜態分析並不需要實際模擬 Android 環境來進行分析，占用的系統資源較少。

William Enck 等人提到[6]了靜態分析需要該應用程式的原始碼才能夠進行分析，但是大部分的應用程式都是封閉原始碼的，並且轉譯技術也無法保證將 bytecode 轉換成原始碼的結果沒有任何錯誤，除此之外靜態分析技術也無法偵測系統的 runtime events、configuration 與執行過程中使用到的檔案，為了能夠檢測出該應用程式是否有不當的利用使用者的隱私資料，William Enck 提出了一個動態敏感性資料偵測系統 TaintDroid，TaintDroid 主要的功能為即時的偵測該應用程式對於敏感性資料的操作，包括陀螺儀、位置訊息、麥克風、相機、書籤和 SMS 等等。透過追蹤敏感性資料的流向，來判斷應用程式是否不當的使用敏感性資料。

Asdroid[7]則提出另一個方法來判斷惡意程式，透過靜態分析的方式來針對程式裡透過背景執行的 SMS 傳輸、HTTP 連線與電話撥號進行掃描，並與 UI 觸發事件進行追蹤，透過語意分析的方式來分析 UI 元件上的標示文字是否正確觸發該功能(如標示”send SMS”的 Button 在點擊之後是否正確觸發 SMS 傳輸的 function)，最後透過動態分析來對應用程式 UI 變化建置 view tree。

Crowdroid [8]採用機器學習的方式來進行分析，Android 為一個 Linux-based 的系統，在 Linux 環境裡 system call 是 user application 與 Linux system kernel 的溝通介面。Crowdroid 透過比較從不同來源所取得的應用程式，在應用程式執行期間，蒐集應用程式所發出的 system call 並進行統計，將這些收集來的統計資料透過 K-mean 集群演算法來進行分析，將應用程式分成無害以及惡意兩個集群。

2.3 Android 自動化測試工具

為了讓應用程式能夠在大多數的 Android 機型上正常運行，開發人員需要將應用程式安裝在每個機型上面，並且針對每個功能進行操作，並收集操作結果進行分析。如果將所有的測試項目全交由人力來進行，將會浪費掉大量的人事成本與時間，透過自動化測試工具，開發者只需要針對待測的測試項目撰寫測試的腳本，並交由自動化測試工具執行，就能夠輕鬆地完成測試，以下將介紹幾種程用的自動化測試工具。

2.3.1 Robotium

Robotium[9]為一個開放原始碼的 Android 自動化測試 Library，是基於 Android 測試框架 Instrumentational 進行的 2 次封裝，將環境設置與一些基本操作進行簡化。Robotium 同時具有黑箱與白箱測試兩種功能，當擁有程式原始碼時，Robotium 可以透過 junit 來對程式碼進行單元測試，當只有該應用程式的 APK 時，Robotium 的黑箱測試可以捕捉到程式執行時畫面上的每個元件(如按鈕、文字元件、選單等等)來進行點擊操作、輸入欄位、滑動手勢、畫面截圖等等。

2.3.2 MonkeyTalk

MonkeyTalk[10]是一套跨平台的開源行動裝置測試工具，支援 iOS 與 Android 應用程式的測試，MonkeyTalk 主要由 MonkeyTalk IDE、MonkeyTalk Agent 還有 MonkeyTalk Scripts 等 3 個部分所組成。在測試開始前需要先將 Agent 植入在待測系統的 應用程式中，之後便可以透過 IDE 來錄製或播放 Scripts，錄製完成的 Scripts 在不同系統之間也可以共用。

2.3.3 UiAutomator

UiAutomator[11]是 google 推出的 UI 測試工具，其功能不限定於應用程式的測試，甚至可對狀態列、webview 及手機上硬體按鈕的操作動作模擬。UiAutomator 進行測試時並不需要待測應用程式的程式原始碼，只要測試行動裝置上有安裝待測應用程式即可。

2.3.4 Monkeyrunner

Monkeyrunner[12]是 Google 提供的一套測試工具，這套工具提供一組指令集，讓使用者可以發送控制命令給 Android 系統的設備。Monkeyrunner 可以應用在多個 Android 裝置控制、功能測試、回歸測試、與延伸自動化。Monkeyrunner 取代人工操作螢幕上可執行的動作，只要讓 adb 與 monkeyrunner 環境可以順利執行，就可以用預先錄製下來的一連串行動 (Scenario)，對 Android 裝置做控制。

2.3.5 Monkey

Monkey[13]是由 google 所提供的一個自動化測試工具，它會向裝置發送隨機的觸碰點擊、拖拉、手勢等等的觸碰輸入事件，可以透過設定亂數 seed 參數值來改變隨機事件的生成方式，常用於應用程式壓力測試，。

2.3.6 Robot Framework

Robot Framework[14]是一個基於 python 的開源自動化測試框架，透過關鍵字驅動的方式來執行測試腳本，測試人員能夠根據測試需求來對關鍵字進行擴充。Robot Framework 具有易於使用的表格式語法、擴充容易、直接支援 Selenium、Java GUI、Telnet、SSH 等平台整合，並提供了一個清楚且直觀的測試結果報表，來方便讓測試人員能夠清楚地了解每個步驟的測試結果。

2.4 Android 爬蟲器

目前的 Android 自動測試工具大部分都需要測試人員自己撰寫測試腳本已進行 Android 應用程式功能的驗證，如 Robotium、UiAutomator、Robot Framework。

Machiry 等人[15]提出了一個 Android 自動化之測試方法，透過 Instrumented SDK 和 Hierarchy Viewer 分析應用程式當下可以觸發的 GUI 事件和系統事件，並且在這些事件中挑選一個事件，透過 Monkey Runner 和 Activity Manager 工具對 Android 裝置進行操作來觸發事件，透過反覆循環這樣的操作流程來達成自動化測試。

SwiftHand[16]是一個透過機器學習的方式來找出擁有較高覆蓋率的最小測試模型，透過記錄每一個操作導致的狀態變化，並且將重複的狀態合併來不斷更新測試模型的狀態機。

2.5 使用工具

2.5.1 DroidBox

Droidbox[15]是 github 上的一個開源的自動化動態測試工具，能夠監控應用程式運行時所產生的 API 呼叫，從而獲取其行為訊息。此工具可偵測行為包括：

- 應用程式訪問網路 Url、IP 的行為資訊。
- 應用程式對檔案讀取及寫入。
- 啟動的服務及程式類別加載、java 的使用。

- 網路外洩的資訊。
- 規避權限。

Droidbox 擁有許多的行為監控項目，但是缺乏對於使用者輸入資料的監控，同時由於 Droidbox 在進行自動化測試的時候使用 Monkey 來對手機進行自動化的操作，容易忽略掉畫面上的元件，導致錯過許多的輸入事件。

2.5.2 Mobile-Security-Framework

MobSF(Mobile-Security-Framework)[18]是一個開源的行動應用程式（Android/ iOS）自動化測試框架，該框架包含靜態與動態分析兩個部分。

靜態分析的部分，MobSF 會對 APK 檔案進行反編譯之後進行分析，收集的項目包括：

- **基本資料分析**

  Package Name、Main Activity、Target SDK、該應用程式所擁有的 Activity、啟用的 Services。
- **權限分析**

  AndroidManifest 所使用到的系統權限與危險性評估、程式碼所使用到的系統權限。
- **程式碼分析**

  使用到的 Android API、程式裡包含的 URL 連結、宣告的變數、APK 檔案裡附帶的檔案。

動態分析的部分，MobSF 會監控應用程式運行過程中的各種行為，包括：

- 應用程式訪問網路 Url、IP 的行為資訊
- 應用程式對檔案讀取及寫入
- 啟動的服務及程式類別加載、java 的使用
- 加密行為
- 使用的 SQLite Databases
- SMS 傳輸

MobSF 在動態分析上的項目較 Droidbox 多且詳細，但是 MobSF 僅提供監控的功能，並不像 Droidbox 包含針對手機的自動化操作，以致於無法達成自動化動態檢測的目的。

2.5.3 SQLite

SQLite 資料庫[19]是一個開放的小型資料庫，近年來在行動裝置系統或是 PC 上的應用開發上越來越常見，它擁有與一般的大型資料庫類似的架構與使用方式(如：MySQL、MS SQL) 應用程式可以建立屬於自己的資料庫，並在資料庫中使用 SQL 基本語法來對資料進行查詢與操作。

3 自動化動態分析系統設計

本研究提出一個自動化的應用程式動態分析測試方法，透過結合 UiAutomator 與 Robot Framework 來進行應用程式的自動化執行與操作，並在應用程式執行期間透過 MobSF 的監控功能來進行應用程式行為資料的收集，最後根據蒐集的資料進行分析。

3.1 敏感性資料的認定

行動應用 App 基本資安規範對於敏感性資料的認定為行動應用程式建立或儲存於行動裝置及其附屬儲存媒介之資訊，而該資訊之洩漏有對使用者造成損害之虞。包含即時通訊訊息、筆記、備忘錄、通話紀錄及簡訊等等資料的洩漏，都有對使用者造成損害的可能，但是 Android 應用軟體的介面設計複雜而且多樣，如 圖 1 與 圖 2，難以進行敏感性資料的分辯與判斷，因此我們統一將使用者的輸入資料作為敏感性資料的目標來作分析。

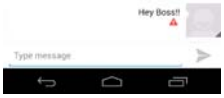

圖 1 簡訊 APP

圖 2 記事本 APP

3.2 自動化執行方法

為了能夠達到完全自動化的測試，並針對測試過程中的畫面資訊進行紀錄，本研究結合 Robot Framework 以及 UiAutomator 測試工具，透過 UiAutomator 來擷取 Android 的畫面資訊進行分析，並透過分析的結果來生成 Robot Framework 的測試腳本，透過自動化的生成測試腳本交由 Robot Framework 執行以達到完全自動化的測試結果，並在 Robot Framework 執行期間針對執行畫面進行搜集，執行流程如圖 3：

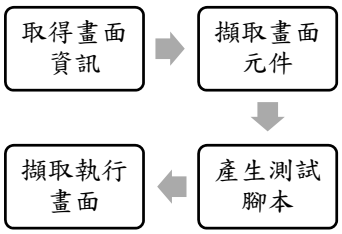

圖 3 執行流程圖

3.1.1 取得畫面資訊

透過 UiAutomator 取得目前手機畫面的資訊，如圖 4 所示：

```
adb shell uiautomator dump
```

```
<node NAF="true" bounds="[0,50][112,146]"
 checkable="false" checked="false" class=
"android.widget.ImageButton" clickable="
true" content-desc="" enabled="true"
focusable="true" focused="false" index="0
" long-clickable="false" package="com.
socialnmobile.dictapps.notepad.color.note
" password="false" resource-id="com.
socialnmobile.dictapps.notepad.color.
note:id/back_btn" scrollable="false"
selected="false" text=""/>
```

圖 4 UI XML 檔案範例圖

node 表示畫面上的元件，node 包含屬性許多屬性，其中 class 表示元件所使用的物件類別，clickable、long-clickable 表示該元件可以被觸碰點擊選取。

3.1.2 擷取畫面元件

為了能夠生成測試腳本，透過擷取 class 為 android.widget.EditText 的元件以及 clickable 為 true 的元件，將元件的 resourceid、class 記錄下來。

```
-------------------------------------------------------------------------------
class:android.widget.EditText
resource-id:com.socialnmobile.dictapps.notepad.color.note:id/edit_title
-------------------------------------------------------------------------------
class:android.widget.ImageButton
resource-id:com.socialnmobile.dictapps.notepad.color.note:id/color_btn
-------------------------------------------------------------------------------
class:android.widget.ImageButton
resource-id:com.socialnmobile.dictapps.notepad.color.note:id/overflow_btn
-------------------------------------------------------------------------------
class:android.widget.EditText
resource-id:com.socialnmobile.dictapps.notepad.color.note:id/edit_note
-------------------------------------------------------------------------------
```

圖 5 擷取結果範例圖

3.1.3 產生測試腳本

本系統採用 resourceid 作為元件的識別，以關鍵字生成 Robot Framework 的測試腳本圖 6，並且執行圖 7。

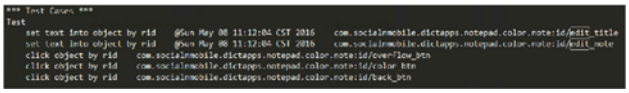

圖 6 腳本範例圖

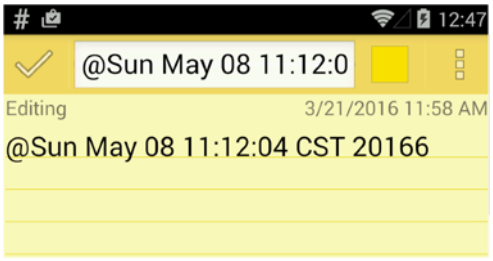

圖 7 執行結果範例圖

3.1.4 擷取執行畫面

在腳本執行期間，系統會針對每次的文字輸入對 Android 畫面進行擷取，並將搜集到的截圖保存起來。

3.2 動態化分析方法

本研究的動態分析主要目的是為了檢測該應用程式是否符合行動應用 應用程式基本資安規範裡的敏感性資料保護項目，透過 MobSF 對於使用者輸入資料的監控與應用程式背景行為的監控來進行比對分析。

3.2.1 前置處裡

透過 AAPT 可以取得應用程式的 XML 設定資料，其中 packagename 與 launchable-activity 將會作為動態測試的參數使用。

```
aapt dump badging <apkfile>
```

3.2.2 啟動虛擬環境

為了能讓測試環境都是處於一樣的狀態，本研究使用 VirtualBox VM 來模擬手機環境，VirtualBox VM 能透過快照來快速建立與還原模擬手機環境的狀態。

MobSF 與手機間的連線是透過 WiFi ADB 來建立，在確定 WiFi ADB 連線成功之後，即可啟用 MobSF 開始進行手機的監控。

3.2.3 啟動自動化執行方法

啟動本文 3-1 所提出的自動化執行方法，開始對應用程式進行探索。

3.2.4 搜集分析結果

MobSF 所蒐集的輸入資料如下，圖 6 是蒐集到的使用者輸入資料，圖 7 是應用程式執行期間讀寫過的檔案清單。

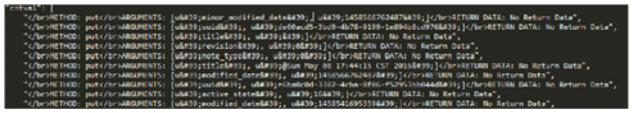

圖 6　使用者輸入資料原始檔案範例

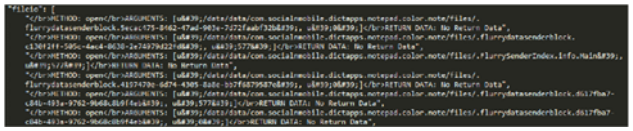

圖 7 檔案清單原始檔案範例

透過 ADB 指令，我們可以從手機取得該應用程式的 SQLite 資料庫。

```
adb pull /data/data/<app_packagename>/databases
```

系統將透過在 AndroidSeeker 執行期間所搜集的畫面輸入資訊紀錄與 MobSF 所蒐集到的 SMS 紀錄、檔案存取紀錄、SQLite 資料庫裡的資訊進行比對，藉此來分析出應用程式如何利用使用者在每個畫面輸入的資料。

四、系統實作

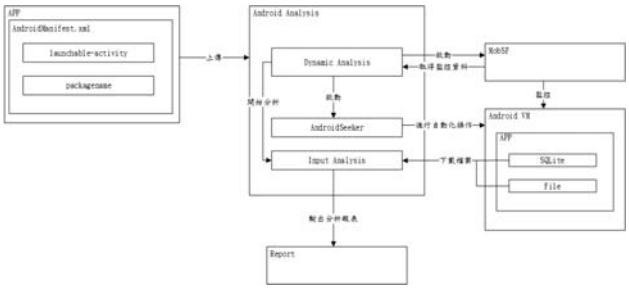

圖 8　系統架構圖

4.1 系統架構

系統主要分成四個部分，以下將會分別說明：

4.1.1 Dynamic Analysis

Dynamic Analysis 主要負責管理整個測試流程，包含接收應用程式並對其進行前置處裡、啟動測試環境、運行 AndroidSeeker、收集 MobSF 的監控資料並交由 Input Analysis 等等。

4.1.2 AndroidSeeker

AndroidSeeker 為本文 3-1 所提出的自動化執行方法之實作，透過 UiAutomator 與 Robot Framework 來對 Android VM 進行各種自動化的操作。

4.1.3 MobSF

MobSF 負責對於手機輸入資料與背景行為的監控。

4.1.4 Input Analysis

Input Analysis 將 MobSF 所蒐集到的與 Android VM 上下載下來的 SQLite 與存取檔案紀錄進行比對，並產生結果報表。

4.2 動態分析測試結果

圖 8 為 ColorNote 的檢測結果，系統將被儲存的資料、位於那個畫面以及儲存的資料格式顯示出來，可以幫助使用者了解在該畫面下的輸入資料被儲存在那個資料表裡。

圖 8　SQLite 檢測結果-以 ColorNote 為例

圖 9 為 chompSMS 的檢測結果，系統將透過 SMS 傳輸的資料內容、傳送的目的地顯示出來，可幫助使用者了解該應用程式是否有私下傳輸使用者的輸入資料。

圖 9　SMS 傳輸檢測結果-以 chomp SMS 為例

五、結論

本論文實作一個 Android 動態檢測平台，以行動應用 App 基本資安規範的敏感性資料保護檢測項目為基礎，透過搜集應用程式在執行期間的行為與畫面紀錄來進行分析，並製作可視化報表，提供資料給使用者了解應用程式對於自己所輸入的敏感性資料的利用，用以判斷該應用程式是否有在使用者不知情的情況下任意使用資料。

致謝

本研究由科技部計畫 MOST 104-2221-E-027-116 所補助，特此感謝。

參考文獻

[1] 行動應用 App 基本資安規範，http://www.communications.org.tw/news/policy/item/8813-20160219.html, Accessed May 20 2016.

[2] The Open Web Application Security Project Mobile Security Project, https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Top_10_Mobile_Risks, Accessed May 19 2016.

[3] NIST Special Publication 800-163 Vetting the Security of Mobile Applications, http://dx.doi.org/10.6028/NIST.SP.800-163, Accessed February 2 2016.

[4] Felt AP, Chin E, Hanna S, Song D, Wagner D. “Android permissions demystified”, Proceedings of the 18th ACM conference on Computer and communications security, pp. 627-638, 2011.

[5] Dietz M, Shekhar S, Pisetsky Y, Shu A, Wallach DS. “Quire: Lightweight provenance for smart phone operating systems.”, Proceedings of the 20th USENIX Security Symp, pp1 347-362 2011.

[6] Enck W, Gilbert P, Chun BG, Cox LP, Jung J, McDaniel P, Sheth AN, “TaintDroid: An information flow tracking system for real- time privacy monitoring on smartphones.”, Communications of the ACM, vol. 57, pp. 99-106, 2014.

[7] Huang J, Zhang X, Tan L, Wang P, Liang B. “Asdroid: Detecting stealthy behaviors in Android applications by user interface and program behavior contradiction.”, Proceedings of the 36th Int’l Conf. on Software Engineering, pp. 1036-1046. 2014.

[8] Burguera I, Zurutuza U, Nadjm-Tehrani S. “Crowdroid: Behavior-Based malware detection system for Android.”, Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, pp. 15-26, 2011.

[9] Robotium, https://github.com/RobotiumTech/robotium, Accessed May 9 2016

[10] MonkeyTalk, http://www.cloudmonkeymobile.com/,

Accessed March 5 2015

[11] UIAutomator, https://developer.android.com/topic/libraries/testing-support-library/index.html#UIAutomator, Accessed June 14 2016

[12] MonkeyRunner, https://developer.android.com/studio/test/monkeyrunner/index.html, Accessed June 14 2016

[13] Monkey, https://developer.android.com/studio/test/monkey.html, Accessed May 22 2016

[14] Robot Framework, http://robotframework.org/, Accessed May 22 2016

[15] Aravind Machiry, Rohan Tahiliani, Mayur Naik, “Dynodroid: An Input Generation System for Android Apps.”, Proceedings of the ACM Symposium on Foundations of Software Engineering, 2013.

[16] Wontae Choi, George Necula and Koushik Sen, “Guided GUI Testing of Android Apps with Minimal Restart and Approximate Learning“, Proceedings of the Object-Oriented Programming, Systems, Languages, and Applications, 2013.

[17] DroidBox, https://github.com/pjlantz/droidbox, Accessed May 9 2016

[18] Mobile-Security-Framework, https://github.com/ajinabraham/Mobile-Security-Framework-MobSF, Accessed April 25 2016.

[19] SQLite, https://www.sqlite.org/index.html, Accessed May 9 2016