Australian Journal of Forensic Sciences

ISSN: 0045-0618 (Print) 1834-562X (Online) Journal homepage: http://www.tandfonline.com/loi/tajf20

Forensic analysis and security assessment of Android m-banking apps

Rajchada Chanajitt, Wantanee Viriyasitavat & Kim-Kwang Raymond Choo

To cite this article: Rajchada Chanajitt, Wantanee Viriyasitavat & Kim-Kwang Raymond Choo (2016): Forensic analysis and security assessment of Android m-banking apps, Australian Journal of Forensic Sciences, DOI: 10.1080/00450618.2016.1182589

To link to this article: http://dx.doi.org/10.1080/00450618.2016.1182589

Australian Journal of Forensic Sciences, 2016
http://dx.doi.org/10.1080/00450618.2016.1182589

Taylor & Francis
Taylor & Francis Group

Forensic analysis and security assessment of Android m-banking apps

Rajchada Chanajitt[a]*, Wantanee Viriyasitavat[a] and Kim-Kwang Raymond Choo[b]

[a]*Faculty of Information Communication and Technology, Mahidol University, Bangkok, Thailand;* [b]*School of Information Technology and Mathematical Sciences, University of South Australia, Adelaide, Australia*

(*Received 29 January 2016; accepted 11 April 2016*)

The increasing popularity and constant evolution of mobile technologies have resulted in an increased focus in mobile device and app security and forensics research. Banking apps appear to be an understudied topic, despite their popularity with consumers. In this paper, seven Android m-banking apps in Thailand are analysed. Based on the findings of our study, we describe the forensic artefacts that could be forensically recovered from the apps, and the findings of security assessment of the apps. For example, we found that several of the apps do not implement root device detection, do not encrypt user data, or it is possible to modify the apps and install repackaged apps.

Keywords: Android app forensics; physical acquisition process; Android app code analysis; Android app repackaging

1. Introduction

In recent years, there has been a significant increase in the number of mobile devices for both personal and professional usage. According to a 2015 survey of smartphone usage, users reportedly spent, on average, 82% of their online time[1] on mobile applications (apps). Mobile apps can be grouped based on their features and functions (e.g. Education, Entertainment, and Social), and implementation approaches (e.g. native, web-based, and hybrid). To be specific, native apps are apps that are designed and built for a specific platform, using libraries and API (application programming interface) to access the data and/or hardware available on the device. A web-based app, on the other hand, is not built for a specific platform but it is hosted on the web and the mobile users access the apps via the browser on the mobile device. Hybrid apps are the combination of native and web-based apps; they are native apps that use a combination of web technologies such as HTML, JSON, and JavaScript. All of the m-banking apps considered in this paper fall into the hybrid app category since the apps are designed for the Android platform and leverage the Android's WebView component to perform necessary financial activities. These hybrid apps are able to access device hardware and data which are usually restricted to access from the browser. Because of this reason, these m-banking apps are able to utilise APIs to retrieve device data and leverage security controls provided by the platform. On the other hand, these apps also have access to sensitive data stored in external storage[2,3] or a SQLite database using Local Data

*Corresponding author. Email: rajchada.ch@acisonline.net

Storage[4]. The improper data protection, thus, may be exploited by an attacker via the hybrid apps (e.g. getting a remote shell on the device or gaining unauthorised access to a backup of the app).

In addition to the nature of the m-banking app implementation, it is of significant importance to ensure the security and privacy of user data, particularly for m-banking apps. This is due to the nature of data and transactions associated with these apps. For example, in 2015, 16.8% of mobile banking (m-banking) users reportedly used smartphones to conduct financial transactions[5]. Common m-banking activities include money transfer and bill payment. In addition, studies have also shown that the prevalence of smart mobile phones is changing consumers' financial behaviour[6]. For example, real-time access to financial information influences users' behaviour on making payments, online shopping and making other financial decisions.

It is, therefore, unsurprising that more financial transactions are made via m-banking apps[6], and the security and privacy of such transactions are considered by users as key concerns[6]. In the survey conducted by the US Federal Reserve[6], 62% of respondents stated that their main reason for not using m-banking was due to security concerns. More than 25% of respondents were also reportedly concerned that their personal information may not be well protected by the mobile apps. Even though security breaches often result in expenses relating to remedial and preventative actions, cost of data loss, and reputational damage, studies have shown that mobile apps are generally not designed with security consideration[6]. As a result, many apps are vulnerable and this enables attackers to gain unauthorised access to the user's sensitive and personally identifiable information (PII)[7].

Due to the constant race between attackers, mobile device manufacturers and mobile app designers, attack trends and vectors are constantly evolving. This is also complicated by the fact that default protection mechanisms provided by mobile device manufacturers varied between manufacturers, when the device was manufactured, whether the operating system or app is up-to-date, whether modifications have been made for a particular service provider, etc. Apps are generally the weakest link in the proverbial saying 'security is only as strong as the weakest link'. For instance, some apps store user credentials in plaintext and store other sensitive and PII on the device unprotected, and it would be trivial for an attacker to use a Java decompiler and reconstruct the code and recover sensitive and PII. In other words, mobile devices are evidentiary sources of criminal investigation, a civil litigation and a corporate disciplinary investigation of employees suspected of breaching company policies. Therefore, it is necessary to have an up-to-date forensic understanding of m-banking and other popular apps[8–12].

A review of the literature also suggests that forensic analysis of m-banking apps appears to be an understudied topic[13–16]. This is the gap that this paper seeks to address. More specifically, in this paper, we study the seven most popular Android m-banking apps in Thailand and our study aims to answer the following research questions.

(1) How much private/personal information is stored on the mobile device after user registration?
(2) How much user and app-generated data are left on the mobile device after banking transactions have been completed?
(3) Can recovered data be used to identify the transactions or activities the user has performed?

(4) Can the app (i.e. code) be modified? If so, can one successfully install modified or repackaged apps on the mobile device?
(5) Can the communication between the apps and the server be intercepted?

To the best of our knowledge, this is one of the first studies on the forensic analysis and security assessment of m-banking apps. The findings and methodologies used in this work will allow forensic practitioners to act and secure user sensitive data in a timely fashion[17,18]. On the other hand, forensic investigators may use the findings and methodologies presented in this paper to reconstruct the chain of events and identify any vulnerability that has been exploited in a security breach.

The paper is organised as follows. In Section 2, we briefly review existing literature on Android devices and app forensics and security assessment research. Section 3 outlines the experimental environment, and the research methodology used in data acquisition and analysis. Section 4 describes relevant findings obtained from the forensic investigation, app code analysis, and app repackaging analysis. Finally, the conclusion is drawn in Section 5.

2. Related work

In this section, we briefly review existing literature on Android device and Android application forensics. While there are several existing studies on both of these topics, studies that are most relevant to our work include the work performed by Immanuel et. al.[4]. In this work, the authors analysed 11 different Android apps and the study reveals that there are significant amounts of user (possibly sensitive) data remaining in the app caches. In addition to the user data, thumbnail images may also be recovered from the Android devices. While the thumbnail images may be fragmented and can only be partially recovered, the images can be used to identify activities that the user has performed on the device. The process to forensically collect, analyse and recover these thumbnails is proposed by Leom et al.[19]. While the aforementioned studies do not focus on the m-banking apps, some of the proposed techniques and methodologies are used in this paper to recover the data or images remaining on the device.

Existing Android forensic studies focused on a range of Android app or app categories. For instance, Mutawa et al.[20] analysed social networking applications, namely Facebook, Twitter, and MySpace. The three apps were installed on three different mobile OS platforms (i.e. BlackBerry, iOS, and Android). Common activities were performed on each app and each device before the logical image of the internal memory of each device was retrieved. The analysis of the obtained images revealed no traces of user activities on the BlackBerry device; however, for iOS and Android devices, the authors could obtain user sensitive information from the acquired images. Another study by Anglano et al.[21] investigated a forensic analysis of the Whatsapp Messenger app on Android smartphones. The investigation was performed on software-emulated Android devices (YouWave virtualization platform) and the acquired image of the internal device memory was in a form of the VirtualBox storage file. The analysis suggested that the WhatsApp Messenger app stores all sent and received messages in the chat database. The database also stores pictures that had been deleted by the user. The work has also shown that the recovered information could be used to reconstruct events and identify when a specific contact was added or deleted and also which and when messages were exchanged and/or deleted. In a more recent work, Yang et al.[22] demonstrated the potential of recovering data remnants from the use of Facebook and Skype

on a Windows 8.1 client machine (e.g. installation or uninstallation of the software, log-in and log-off information, contact lists, conversations, and transferred files).

In addition to social networking apps, another app category that may store sensitive user information is the healthcare app. In 2015, Azfar et al.[17] investigated 40 popular free Android health apps. Their study revealed that the majority of the apps store private and sensitive user details, user activities information, and timestamp information in the local databases. Such information, once recovered, can be used to reconstruct the user activities and timeline as well as the user's whereabouts. We were only able to locate one publication that examined m-banking apps, which is most relevant to our work. Jung et al.[23] analysed seven Android m-banking apps in South Korea with the aims of identifying vulnerabilities and proposing technical countermeasures. More specifically, they decompiled the apps to obtain the smali code and attempt to perform a repackaging attack on these apps. Their findings revealed that APK hashing and Anti-virus checking mechanisms could be easily bypassed and repackaged apps could be run successfully without the need to obtain any of the sender's personal information (e.g. sender's public key certificate, bank account password, or security card). As a result, this finding implies that banking transactions can be hijacked and the fund transfer can be directed to the attacker's account. Despite the vulnerabilities presented in the paper, the work does not focus on the forensic analysis of the apps, which is one of the main focuses of this paper.

3. Experiment setup

3.1. Equipment used in this experiment

Two Android devices, namely a Samsung Galaxy S4 GT-I9500 (rooted) and Galaxy S4 mini GT-I9190 (unrooted), were used as the test devices. Both devices used Android Kitkat 4.4.2 version. In order to compare the impact of different settings of the device, we performed forensic investigation on the unrooted (i.e. Samsung Galaxy S4 mini GT-I9190 device) and Samsung Galaxy S4 GT-I9500 device, which was rooted and configured with the Android developer mode (ADB) and the USB debugging mode enabled. Storage encryption software was not installed on the devices, and seven popular Android m-banking apps in Thailand (see Table 1) were then installed on both devices. For the purpose of the experiments, a user account was created on each

Table 1. Android m-banking apps used in the study.

		App Functions				
App name	App version	Balance inquiry	Money transfer	Calendar/ reminder setting	Branch/ATM location search	Bill payment
Bank A	V2.5.1	Yes	Yes	Yes	Yes	Yes
Bank B	V1.0.91	Yes	Yes	No	Yes	Yes
Bank C	V2.7.1	Yes	Yes	No	Yes	Yes
Bank D	V1.4.0	Yes	Yes	No	Yes	Yes
Bank E	V1.2.1.2	Yes	Yes	No	No	Yes
Bank F	V6.12	Yes	Yes	No	Yes	Yes
Bank G	V1.3.3	Yes	No	No	Yes	No

m-banking app and typical user activities were conducted. The activities performed on each device are listed in Table 2. For image acquisition and forensic analysis, we used a number of forensic tools such as JTAG and Android SDK (for image acquisition) and Autopsy and FTK Imager (for forensic analysis).

3.2 Research methodology

3.2.1. Stage 1: Android forensic memory acquisition and analysis

After the above activities were performed on the m-banking apps, we acquired a forensic image from the device. In this paper, two acquisition methods were used, namely: DD and JTAG. The DD method creates a bit-by-bit copy of the entire flash memory to an external SD Card. However, this method can only be used on the rooted device. Thus, we used the DD method to acquire the image from the Galaxy S4 device. In the case of the Galaxy S4 mini, which is not rooted, the image was acquired via the JTAG pins. Because JTAG is a physically invasive method, it does not require USB debugging to be enabled on the device and can access the (non-rooted) device with root permission. In addition, it can circumvent passcode lock controls on the device by bypassing the operating system. However, this method requires the phone to be physically disassembled in order to connect the JTAG box with the JTAG pads on the phone. The location of the JTAG pins may vary with the phone model. The JTAG pins of the Galaxy S4 mini are depicted in Figure 1, and the pins can be identified when the clips securing the plastic midframe of the device is removed.

The forensic acquisition process used in the study is described below and the flow-chart of the process is presented in Figure 2.

(1) Once a mobile device is acquired, it is necessary to determine whether the device is powered on or off.

- If the device is powered on, we check whether the device is locked.
  - If the device is unlocked, one could acquire a logical forensic image using the necessary tools.
  - Otherwise, a screen lock needs to be bypassed.
- In the case that the device is powered off, we have to check whether a physical acquisition via JTAG can be performed (based on the SWGDE methodology[24]).
  - If it can be performed, we will use the JTAG method described above to obtain the image.
  - Otherwise, the device is powered on. In case the device is rooted and USB debugging is enabled, the DD method can then be used to acquire data on the rooted device or use a logical acquisition method if the device is not rooted.

To obtain a forensic image via the DD command, the device is first connected via the Android Debug Bridge (ADB) shell. We then acquire the root privilege and locate the data using the mount command. After locating the partition, we use the DD commands shown below to acquire the bit-by-bit copy of the entire flash memory.

'dd if=/dev/block/mmcblk0 of=/mnt/sdcard/data.img bs=4096 conv=noerror,sync'.

(2) This command allows us to make a copy of the entire flash memory and saves it to the external storage (SD card) for further analysis.

Table 2. Summary of user activities performed on the seven m-banking apps.

Transaction date (day/month/year)	Smartphone device	Transaction type	Transaction description
03/07/15	GalaxyS4 GT-I9500	Money transfer	Transfer 1000 Baht from the Bank E account to the Bank A account
			Transfer 1000 Baht from the Bank B account to another Bank B account
			Transfer 1000 Baht from the Bank C account to the Bank B account
		Bill payment	Mobile bill payment to AIS from the Bank C account for an amount of 666.61 Baht
		Balance inquiry	Check of balance of the Bank B account
			Check of balance of the Bank F account
			Check of balance of the Bank E account
			Check of balance of the Bank G account
			Check of balance of the Bank A account
			Check of balance of the Bank D account
			Check of balance of the Bank C account
10/08/15		Bill payment	Internet bill payment to AIS from the Bank A account via Scan Bill for an amount of 1273.30 Baht
11/08/15		Bill payment	Mobile bill payment to AIS from the Bank E account for an amount of 132.43 Baht
		Money transfer	Transfer 5000 Baht from the Bank D account to another Bank D account
		Branch location	Search for ATM/Branch from the Bank A account
			Search for ATM/Branch from the Bank B account
			Search for ATM/Branch from the Bank D account
			Search for ATM/Branch from the Bank C account
			Search for ATM/Branch from the Bank G account
			Search for ATM/Branch from the Bank F account
14/09/15		Bill payment	Mobile bill payment to AIS from the Bank F account for an amount of 135.64 Baht
			Mobile bill payment to AIS from the Bank D account with 886.64 Baht
21/09/15		Bill payment	Internet bill payment to AIS from Bank B account via Scan Bill for an amount of 1273.30 Baht
25/09/15		Money transfer	Transfer 1000 Baht from the Bank A to the Bank F account
			Transfer 1000 Baht from the Bank F account to another Bank F account
		Reminder setting	Setting General Reminder at 11:00 AM to 12:00 PM on the Bank A account

(*Continued*)

Table 2. (*Continued*).

Transaction date (day/month/year)	Smartphone device	Transaction type	Transaction description
22/09/15	GalaxyS4 GT-I9190	Money transfer	Transfer 5000 Baht from the Bank D account to another Bank D account
29/09/15		Balance inquiry	Check for balance of the Bank E account
14/10/15		Bill payment	Mobile bill payment to AIS from the Bank F account for an amount of 1045.39 Baht
			Mobile bill payment to AIS from the Bank B account for an amount of 132.43 Baht
		Balance inquiry	Check balance of the Bank F account
			Check balance of the Bank B account
20/10/15		Bill payment	Internet bill payment to AIS from the Bank D account via Scan Bill for an amount of 1273.30 Baht
		Balance inquiry	Check balance of the Bank D account
			Check balance of the Bank A account
		Money transfer	Transfer 1000 Baht from the Bank B account to another Bank B account
		Branch location	Search for ATM/Branch from the Bank D account
			Search for ATM/Branch from the Bank B account
			Search for ATM/Branch from the Bank A account
01/11/15		Money transfer	Transfer 2000 Baht from the Bank F account to the Bank E account
			Transfer 1000 Baht from the Bank E account to the Bank A account
			Transfer 1000 Baht from the Bank A account to Bank G account
			Transfer 375 Baht from the Bank C account to another Bank C account
		Balance inquiry	Check balance of the Bank C account
			Check balance of the Bank G account
		Bill payment	Internet bill payment to AIS from Bank D account via Scan Bill for an amount of 1273.30 Baht
			Mobile bill payment to AIS from Bank A account for an amount of 140 Baht
			Mobile bill payment to AIS from Bank C account for an amount of 745.26 Baht
		Reminder setting	Setting General Reminder at 21:05 to 21:07 on the Bank A account
		Branch location	Search for ATM/Branch from the Bank C account
			Search for ATM/Branch from the Bank G account
			Search for ATM/Branch from the Bank F account

(3) To obtain a forensic image via the JTAG method, one needs to have the technical expertise and training to extract and analyse binary images in a forensically sound manner. While this method allows one to directly access memory chips

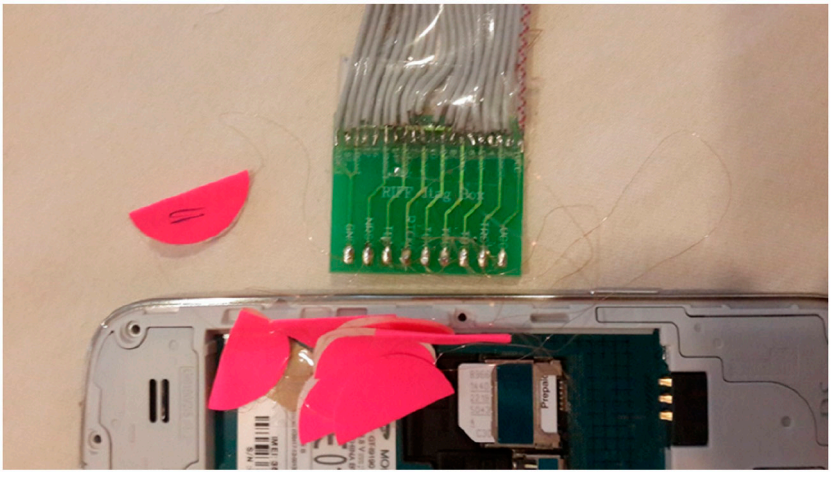

Figure 1. JTAG pins on Samsung Galaxy S4 mini.

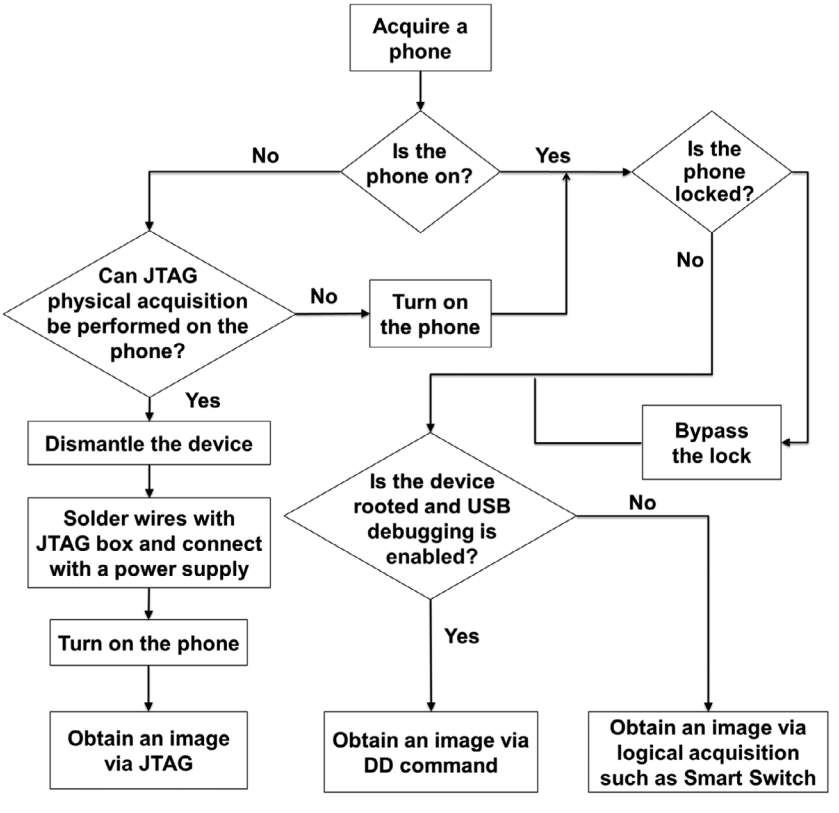

Figure 2. Physical acquisition process.

without requiring the user's password, it requires physical disassembling of the device, probing the JTAG Test Access Ports (TAPs) for the proper connection, and soldering connectors to the JTAG ports. In addition, since JTAG pads on the mobile devices vary from model to model, finding the correct JTAG pads can be challenging.

After obtaining a forensic image via the acquisition process, forensic examination and analysis were performed on the acquired image of each device. The analysis was performed on a forensic workstation (in our experiments, we used a Macbook Pro and a Windows 7 personal computer). In our experiments, the analysis was manually performed using open source forensic tools, namely: Autopsy and FTK Imager. More specifically, we investigated the app folders in the data partition to determine whether any use activities and user-specific data could be recovered. In addition, the cache and system partitions were analysed to locate cache or temporary files used by the apps. Images and media files that are relevant to the apps were determined to be stored in the /data/media folder. The unallocated space was also investigated for deleted data that could be recovered. We further investigated the information stored in subfolders in the /data/data/[app_package] folder. A list of subfolders investigated is shown in Figure 3.

3.2.2. Stage 2: Android app code analysis

In addition to the forensic image analysis, we performed application code analysis to evaluate the security features, implemented by the apps. The application code was decompiled and analysed using Java decompiler. This allows us to identify how configuration entries are created, how the app databases are populated and read, and how security features (i.e. signature verification and certificate pinning) are implemented and used. The application code analysis is vital since the attacker can exploit an app's vulnerability by analysing the app function or recompiling the package with embedded malware even though the app does not store user's sensitive information.

In order to determine the security feature of the app, the application code/binaries were statically analysed. Android apps are generally packaged and distributed as .apk files. The .apk files can be extracted from the forensic image and renamed to .zip file. The latter can be unzipped in order to view and retrieve sub-directory files. These sub-directory files generally contain all compiled resources (compiled code, images, layouts, xml files, databases, etc.) required by the app. Then, Dex2Jar, apktool, and JD-GUI are utilised to reconstruct the application code for further analysis.

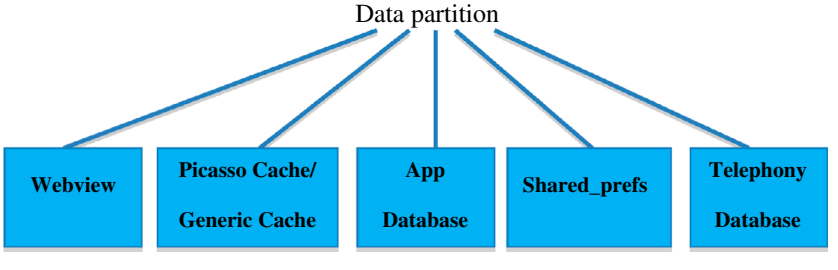

Figure 3. Forensic artefacts categorised by data folder located in user data partition (/data/data/[app_package folder]).

Table 3. Forensic artefacts recovered from the app_package folder in data partition using DD and JTAG physical extraction.

App name	WebView	Picasso/generic cache	App database	Shared_prefs/files	Telephony database
Bank A	Cookie	-	ATM location, branch location, reminder	registerID, user_image	Ref:3122 OTP:933,549 OTP will expire within five minsType: InterBank From: 3,951,353,254 To: 020,146,963,580 Amt: 2000.00 Ref.No.: BAYM11964882
Bank B	N/A	Thumbnail (transfer, payment)	Encrypted PIN_Lock and PASS_Code	registerID	Transfer to a/c of MALEE CHANACHIT <x009133>with <OTP 127,912> within 5 min
Bank C	-	Thumbnail (transfer, payment)	-	User_picture, email address, mobile operator, app_configuration	-
Bank D	-	-	-	Encrypted key, app device token	K.rajchada transferred 5000.00Bt to your A/c 1181xxx765
Bank E	N/A	User_image	-	citizenID, user_picture (/media/0/Bank E/ kitty.png), sessionID	-
Bank F	Account number, balance, account type (file_0.local storage)	Transaction_image cache, thumbnail	-	citizenID, credit card number, date of birth, address, phone number, email address	TO*p*=888,159 Ref20227446-ADD 3rd A/C น.ส.ดวงเดือน บัวบุญ 9824XXXXX5 Funds Transfer Ref.23540262992015925: 17,309.16 BHT to 9824XXXXX5 was successful
Bank G	-	-	Cleartext USERPIN (localstorage.db-journal)	-	-

3.2.3. Stage 3: Android repackaging analysis

In the last stage, the apps were repackaged to interrupt certificate checking. These modified apps were signed by our locally generated keypair using the jarsigner tool and were installed on the device (this is the same approach used by researchers, such as Do, Martini and Choo[25]). Similar to the approach undertaken by D'Orazio and Choo[26], we used the Burp proxy certificate to intercept traffic between the apps and the server. Moreover, the Android built-in 'logcat' command was run while the apps were running to analyse whether any sensitive data were sent during the runtime.

4. Findings

4.1. Stage 1: Forensic artefacts via physical image analysis

Our investigation of both Android devices indicated that Usagestats in the system folder and media folder of data partition contain data of forensic interest. For example, we recovered the financial transactions generated by the mobile users and the associated timestamp (recorded in Unix Epoch format). We also determined that Bank B, Bank C, and Bank F apps save the receipt of a money transfer transaction as a JPEG file. Each app_package in data partition was investigated and the results are shown in Table 3.

In the app_package folder, the shared_prefs stores sensitive user information such as citizenID in Bank E and Bank F. The cache did not appear to contain sensitive information. To analyse the SMS message and information relating to a sent SMS message (e.g. from a bank to a client at the conclusion of a transaction), we examined the 'mmssms.db' database. From the recovered message (content and timestamp), we were able to confirm the occurrence of the financial transactions. For example, from the recovered SMS message, we obtained the one time password (OTP) provided by the m-banking institution to verify the transaction. We were also able to obtain the encrypted tokens used to authenticate the user. The forensic artefacts from Autopsy and FTK imager are compared and adb logcat to view the log messages as shown in Table 4.

4.2. Stage 2: Findings on app code analysis

In our analysis, we determined that Bank A, Bank B, Bank C, and Bank E, use the Secure Random method to initialize the cryptographic keys and AES algorithm to protect data in transit. Both Bank A and Bank B use binary code obfuscation to complicate reverse engineering efforts. All apps were determined to be digitally signed with a valid RSA public key and SHA hashing algorithm. SSL pinning is implemented on Bank A, Bank D and Bank E, and Hostname Verification adopted by Bank E. For Bank A and Bank D, these two apps have a hard-coded public key. For apk signature, PackageManager.PackageInfo flag is used which returns information about the signature included in the package. Bank B, Bank C, Bank D, and Bank E implement root device detection when the user launches the app. The findings are shown in Table 5.

Our study revealed that Bank E does not load a third-party library to run the app and goes into idle mode immediately once the device's screen is off. This is different from the other apps, which have a session timeout between five and fifteen minutes. Meanwhile, Bank D is the only app that calls web services. In the event that a

Table 4. Comparison between forensic acquisition methods and logcat debugging.

App name	Physical acquisition methods		
	JTAG	DD	ADB logcat
Bank A	-	Userimage_1136177018.dat (user picture)	RegID, Search Location, account number, Server URL
Bank B	Com.google.maps.api.android.lib6.drd.PREFERENCES_FILE.xml (SessionID)	-	sessionID and Path=/Bank B
Bank C	No_backup_folder, wap_PreferenceForKong.xml (Mobile operator)	-	-
Bank D	DATA_Tiles-2 (device_uuid)	-	-
Bank E	DEPreferences.xml.bak (user picture)	-	Server URL, citizenID, uniqueKey
Bank F	analytics.log.0.lck (user,ack), wl.log.0.send (user,ack)	file_0.localstorage (Acct Number, Acct Balance)	Last login date/time, userID, sessionKey, Customer name, account number, account balance
Bank G	-	-	SSL certification check URL

Table 5. Application code analysis for security mechanisms on Android rooted device.

App name	Root device detection	Encryption applied	Certificatepinning	Obfuscationapplied	Signature algorithm
Bank A	No	AES/CBC/PKCS5Padding	SSL Pinning with hard-coded public key	Yes	SHA256withRSA
Bank B	Yes	AES/CBC/PKCS5Padding	SSL Pinning with Hostname Verification	Yes	SHA256withRSA
Bank C	Yes	AES/CBC/PKCS7Padding	No (has MyTrustManager and SSLUtils but not use)	No	SHA256withRSA
Bank D	Yes	No (has SimpleAESCrypt class but not use)	SSL Pinning with hard-coded public key	No	SHA1withRSA
Bank E	Yes	AES/CBC/PKCS7Padding	SSL Pinning with Hostname Verification	No	SHA256withRSA
Bank F	No	No (has SecurityUtils class but not use)	No (has WLAuUserAuthManager and HttpClientManager class but not use)	No	SHA1withRSA
Bank G	No	No	No (has package com.trusteer.otrf.f and package ny0 k but not use)	No	SHA256withRSA

malicious user managed to obtain the login credential used on the app, it can be used to access the banking account via a browser directly. In addition, since activity and receiver components are usually shared (i.e. they have intent filters or export attributes set), these components can be a potential attack vector to exfiltrate app data to the unauthorised apps.

4.3. Stage 3. Findings on app repackaging analysis

After conducting the app code analysis, the apps were modified, repackaged and signed with the author's certificate. In order to install the modified apps of Bank A, Bank D, and Bank E on the rooted device, we need to uninstall the existing app before installing the modified one. In addition, the SSL pinning that is used to verify the certificate needs to be bypassed. To do this, the checkServerTrusted method implemented in the code was commented out. The reinstallation and SSL pinning bypass were successfully performed only on Bank D app and the sensitive user data such as PIN code could be intercepted. On the other hand, Bank C, Bank F, and Bank G do not require the

Table 6. Application repackage analysis and SSL interception on Android rooted device.

App name	SSL intercept without app installation	SSL intercept with app installation	Reuse auth token on another device
Bank A	-	A random AES key each for login via requestkey parameter. While forwarding the POST request, the Internet Connection error message appeared on the screen which means the installed certificate cannot be used	No auth_token (AES key instead)
Bank B	Unable to intercept traffic	–	No auth_token (only cookie)
Bank C	Random two auth_tokens, and public key	–	Unable to reuse auth_token, and received the 'Session Timeout' error
Bank D	–	SSL pinning is bypassed and banking transactions as well as PIN code are shown in cleartext,	Able to use the same sessionID to login to app while another device is using this session
Bank E	–	Hostname Verification accompanied with SSL pinning; however, the app cannot be started	N/A
Bank F	Able to view banking transaction history, userID and password in cleartext	–	N/A
Bank G	Cleartext password, Customer name, phone number, tokenID, encryptedKey, encryptedSecretKey, account ID, ATM latitude and longitude	–	Dynamic token and to use the app on another device, it requires that the app has been previously activated

uninstallation prior to installing the modified apps. These apps allow a third-party certificate (in our context, Burp CA Certificate) to import into the device and this becomes the user's trusted credential. The communications of these three apps can be monitored using a proxy and we observed that one can view the sensitive user data from the POST method request and server response. The detailed results are described in Table 6.

As shown in Table 6, we were unable to bypass the SSL pinning on Bank E and Bank A apps. For example, Bank E uses the checkSystemTrust and checkPinTrust method, and Bank A verifies an unexpected public key in the checkServerTrust method. As a result, we were not able to intercept the traffic sent by these apps. However, for the Bank D app, we were able to bypass SSL pinning and recovered sensitive information sent in plaintext during transit.

5. Conclusion

While mobile security risks may be seen by some as an extension of existing threats to traditional computing systems, the mobile threat landscape is an extremely fast-moving environment. As noted by Imgraben et al.[27], mobile security risks are 'not just to the owner of the smart mobile devices, but also to the organisations they work for and to other organisations connected to the Internet'. It is, therefore, essential for security and forensic researchers and practitioners to keep pace with technologies, including mobile technologies.

In this paper, we conducted forensic investigations and security assessment analysis of seven popular Android m-banking apps in Thailand. We determined that data of forensic interest, including sensitive user data, pertaining to the use of such apps can be recovered using DD and JTAG techniques. For example, we were able to recover from Bank F information such as account number, account type, and account balance, from Bank B, Bank C, and Bank F information such as citizenID, date of birth and thumbnails for banking transactions, and from Bank A and Bank G information such as user's PIN code. The SMS messages we recovered could also be used to verify prior transactions (e.g. timestamp and OTP).

We also conducted a security assessment analysis of the apps, and determined that more than three of the apps investigated do not implement root device detection. In addition, despite the built-in encryption libraries, the study reveals that some apps do not encrypt user data. The repackaging app analysis also shows that it is possible to modify the m-banking apps and install the repackaged apps. If the repackaged apps are implemented to intercept SSL traffic, an attacker is then able to intercept SSL traffic and obtain sensitive information. This vulnerability is observed in more than half of the m-banking apps considered.

Our findings echoed the recommendations of D'Orazio and Choo[7] who posited the importance of introducing security into the design of the apps. For example, app developers need to understand how the app functions and have the capability or access to security testing (e.g. using OWASP guidelines[28]) prior to releasing the app. In addition, the apps should not store cleartext credentials and information used by the app should be encrypted[29].

Disclosure statement

No potential conflict of interest was reported by the authors.

References

1. 82% of mobile media time is via Apps [Internet]. Bosomworth D: Flurry Analytics [cited May 2012]. Available from: http://www.smartinsights.com/mobile-marketing/app-marketing/82-of-mobile-media-time-is-via-apps/
2. Do Q, Martini B, Choo K-KR, editor. Enhancing user privacy on android mobile devices via permissions removal. Proceedings of the 2014 47th Hawaii International Conference on System Sciences (HICSS); 2014 Jan 6–9; Hawaii, USA.
3. Do Q, Martini B, Choo K-KR, editor. Enforcing file system permissions on android external storage: Android File System Permissions (AFP) prototype and ownCloud. 13th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom); 2014 Sep 24-26; Beijing, China.
4. Immanuel F, Martini B, Choo K-KR, editor. Android cache taxonomy and forensic process. Proceedings of 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom); 2015 Aug 20–22; Helsinki, Finland.
5. Mobile Banking From Innovative Trend to Staple Service? [Internet]. Sugarloaf Parkway (GA): Georgia Credit Union Affiliates [cited 2013]. Available from: http://www.gcua.org/publications/considerthis/2013/jul/jul13.html
6. Consumers and Mobile Financial Services 2015 [Internet]. Office of Inspector General (WA): Board of Governors of The Federal Reserve System [cited 2015 March]. Available from: http://www.federalreserve.gov/econresdata/consumers-and-mobile-financial-services-report-201503.pdf
7. D'Orazio C, Choo K-K R, editor. A generic process to identify vulnerabilities and design weaknesses in iOS healthcare apps. Proceedings of the 2015 48th Hawaii International Conference on System Sciences (HICSS); 2015 Jan 5–8; Hawaii, USA.
8. Daryabar D, Dehghantanha A, Eterovic-Soric B, et al. Forensic investigation of OneDrive, Box, GoogleDrive and Dropbox applications on android and iOS devices. Australian Journal of Forensic Sciences. [In press, DOI: http://dx.doi.org/10.1080/00450618.2015.1110620].
9. Norouzi F, Dehghantanha A, Eterovic-Soric B, et al. Investigating social networking applications on smartphones: detecting Facebook, Twitter, LinkedIn, and Google+ Artifacts on android and iOS platforms. Australian Journal of Forensic Sciences. [In press, DOI: http://dx.doi.org/10.1080/00450618.2015.1066854].
10. Shariati M, Dehghantanha A, Choo K-KR. SugarSync forensic analysis. Australian Journal of Forensic Sciences. 2015;48(1):95–117.
11. Martini B, Do Q, Choo K-K R. Chapter 15 – Mobile cloud forensics: an analysis of seven popular Android apps. In Ko R and Choo K-K R, editors. Waltham, MA: Syngress, an Imprint of Elsevier; 2015. Cloud security ecosystem; pp. 309–345.
12. Do Q, Martini B, Choo K-KR. A forensically sound adversary model for mobile devices. PLOS ONE. 2015;10(9):e0138449.
13. Azfar A, Choo K-KR, Liu L. Android mobile VoIP apps: a survey and examination of their security and privacy. Electronic Commerce Research. 2016;16(1):73–111.
14. Barmpatsalou K, Damopoulos D, Kambourakis G, et al. A critical review of 7 years of mobile device forensics. Digital Investigation. The International Journal of Digital Forensics & Incident Response. 2013;10(4):323–349.
15. Sufatrio, Darell JJ Tan, Tong-Wei Chua, et al. Thing: securing android: a survey, taxonomy, and challenges. ACM Computing Surveys (CSUR). 2015;47(4):58.
16. Polla LM, Martinelli F, Sgandurra D. A survey on security for mobile devices. IEEE Communications Surveys and Tutorials. 2013;15(1):446–471.
17. Azfar A, Choo K-K R, Liu L, editor. Forensic taxonomy of popular android mHealth apps. Proceedings of 21st Americas Conference on Information Systems (ACMIS); 2015 Aug 13–15; Puerto Rico, USA.
18. Azfar A, Choo K-K R, Liu L. An android communication app forensic taxonomy. Journal of Forensic Sciences [In press, accepted 5 October 2015].
19. Leom MD, D'Orazio C, Deegan G, et al., editor. Forensic collection and analysis of thumbnails in android. Proceedings of 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom); 2015Aug 20–22; Helsinki, Finland.
20. Mutawa AN, Baggili I, Marrington A. Forensic analysis of social networking applications on mobile devices. Digital Investigation. 2012;9: S24–S33.

21. Anglano C. Forensic analysis of WhatsApp Messenger on android smartphones. Digital Investigation: The International Journal of Digital Forensics & Incident Response. 2014;11 (3):201–213.
22. Yang TY, Dehghantanha A, Choo K-KR, et al. Windows instant messaging app forensics: Facebook and Skype as case studies. PLOS ONE. 2016;11(3):e0150300.
23. Jung J, Kim JY, Lee HC, et al. Repackaging attack on android banking applications and its countermeasures. Wireless Personal Communications. December 2013;73(4):1421–1437.
24. SWGDE Best Practices for Examining Mobile Phones Using JTAG Version: 1.0. Scientific working group on digital evidence [cited 2015 September 29]. Available from: https://www.swgde.org/documents/Current%20Documents/2015-09-29%20SWGDE%20Best%20Practices%20for%20Examining%20Mobile%20Phones%20Using%20JTAG
25. Do Q, Martini B, Choo K-KR. Exfiltrating data from android devices. Computers & Security. 2015;48:74–91.
26. D'Orazio C, Choo K-KR. An adversary model to evaluate DRM protection of video contents on iOS devices. Computers & Security. 2016;56(C):94–110.
27. Imgraben J, Engelbrecht A, Choo K-KR. Always connected, but are smart mobile users getting more security savvy? A survey of smart mobile device users. Behaviour & Information Technology. 2014;33(12):1347–1360.
28. Application Security Guide For CISOs Version 1.0 [Internet]. Open Web Application Security Project (OWASP) [cited 2013 November]. Available from: https://www.owasp.org/images/d/d6/Owasp-ciso-guide.pdf
29. 2015 State of Application Security: Closing the Gap [Internet]. Bird J, Johnson E, Kim F: SANS Institute InfoSec Reading Room [cited 2015 May]. Available from: https://www.sans.org/reading-room/whitepapers/analyst/2015-state-application-security-closing-gap-35942