FORENSIC SCIENCE JOURNAL SINCE 2002

Forensic Science Journal
2017;16(1):1-12
fsjournal.cpu.edu.tw
DOI:10.6593/FSJ.2017.1601.01

A Study on Common Android Emulators and Anti-Forensic Message-Hiding Applications

Szu-Yuan Teng [1*], M.S. ; Che-Yen Wen [2], Ph.D.

[1] *Taipei City Field Office, Investigation Bureau, Ministry of Justice*
[2] *Department of Forensic Science, Central Police University*

Received: December 23, 2016; Accepted: May 22, 2017.

Abstract

Currently, mobile devices are widely used in various walks of life. The Android operating system has the highest market share of the mobile devices operating system market. Android can be installed in physical mobile devices; however, Android mobile operating system emulators are also available. Users can install applications (APPs) in an emulator for convenient use without physical mobile devices. There are several message hiding APPs (e.g., Wickr) that provide end-to-end encryption and message self-destruction mechanisms. Criminals can use these message hiding APPs, with their anti-forensic features, to send secret messages. These message-hiding APPs, installed in an Android emulator to evade criminal investigation, make digital forensics very challenging. Investigators need to know how criminals install and use such emulators in physical devices, how criminals install and use message-hiding APPs in the emulator, and how messages can be. This study explores applications of digital forensic tools and forensic procedures to identify and analyze four message hiding APPs installed in emulators: Wickr, Surespot, Cyber Dust, and ChatSecure. The emulators used in the study are AMIDuOS, Andy, BlueStacks App Player, Droid4X, Genymotion, KOPLAYER, Memu, Nox App Player, Windroy, Xamarin Android Player, and YouWave Android. Their forensic signatures and application characteristic values are sorted and summarized for digital forensics, so that digital forensic personnel can refer to this digital forensic method when analyzing criminal evidence using an Android emulator.

***Keywords:** mobile device forensics, Android emulator forensics, anti-forensics, message hiding application, message hiding, application forensics*

Introduction

Android and iOS are presently the two most common mobile operating systems. According to the IDC 2015 Smartphone OS Market Share Report [1], Android has the largest market share at 81.2%, followed by iOS at 15.8%. The Android mobile operating system can be installed in physical mobile devices; however, Android emulators are available for users to install and use Android applications.

The main function of an Android emulator is to simulate the software and hardware environment of an Android mobile device (e.g., mobile phone or tablet). Emulators are typically used to enable PC users to download and use applications (e.g., games) from the Google Play store. Android emulators include the complete Android architectures, including the Linux Kernel, Native Library, Dalvik VM, and Android application framework.

In recent years, personal information leakage and other information security incidents have occurred with increasing frequency. Thus, demand for privacy protection is rising. In response, several real-time communication software vendors with the highest popularity and the largest user quantity have started providing end-to-end encryption. For example, the WhatsApp application

[*]Corresponding author: Szu-Yuan Teng, Taipei City Field Office, Investigation Bureau, Ministry of Justice.
E-mail: mjib.teng@gmail.com

protects user content (e.g., text messages, photos, videos, group chats, and video chats) with end-to-end encryption. Only the sender and the recipient can read the encrypted messages. The Line instant communications application has an end-to-end encryption capability called “Letter Sealing”. It applies by default to all chat, voice and video calls. Because messages are compiled into messy code using encryption keys stored in the personal devices rather than on the server, any message that is intercepted cannot be decrypted and read. The message-hiding application Telegram became infamous after Islamic State (ISIS) terrorists used it to exchange messages about a terrorist attack in Paris on Friday 13 November 2015. It has an automatic message destruction function to reduce the risk of being monitored. Such an application can be used as an instrument of crime because of its strong privacy protections. ISIS has developed another encrypted communication application called "Alrawi," which increases the difficulty of spying on their terrorist activities by anti-terrorism units. The United States government has warned that criminals and extremists could use such communication encryption technology to hide their whereabouts.

Suspects are likely to use Android emulators to install such message-hiding applications for criminal message transmission with the purpose of evading criminal investigation. Digital forensics practice personnel therefore need an in-depth discussion and study of how to analyze criminal cases that involve the use of new virtual mobile devices as instruments of crime.

Experimental Materials and Methods

Android Emulator Software

In this study, an experiment was conducted on 11 Android emulators. The Lollipop version of the AMIDuOS emulator was used as the research subject [2]. The Andy emulator can run on the Microsoft Windows operating system and the Apple OS X operating system. It has powerful functions, and it supports seamless synchronization between desktop and mobile devices [3]. The BlueStacks App Player is one of the earliest Android emulators in the market, and it is one of the most famous and most widely used emulators [4]. Droid4x, also known as the hippocampus-playing simulator, enables ARM applications to run on an x86 architecture, and it is compatible with more than 99 % of applications and games in the market [5]. The Genymotion Android Emulator claims to be the Android emulator software with the fastest starting speed, and it currently supports operating systems including Microsoft Windows, Apple OS X, and Linux, with the features of being easy to install and use [6]. The KOPLAYER emulator, developed by Kaopu Network Co., Ltd. in Fuzhou China, supports Intel and AMD CPUs [7]. The Memu emulator, developed by Microvirt Software Technology Co., Ltd. in Shanghai of mainland China, provides a multiple boot manager function like Droid4X [8]. The Nox App Player, an emulator developed by MoreTech Inc. in Beijing, China, emphasizes high performance and ultimate compatibility [9]. The Windroy emulator was developed by Beijing Windroy Technology Co., Ltd. in mainland China [10]. Xamarin Android Player, an emulator developed by the company Xamarin, can be installed on Microsoft Windows and Apple OS X. It is mainly intended for use by application developers [11]. YouWave Android, a commercial emulator developed by the company YouWave in California, United States, supports Android 5.1 Lollipop version [12]. This study also selected 11 types of common Android emulators in the market as experiment and analysis objects, including Andy v46.2.207, AmiDuos v3.1.30, BlueStacks App Player v2.0, Genymotion v2.6.0, Memu v2.6.5, Droid4X v0.10.3, KOPLAYER v1.3.14, Nox App Player v3.1, Windroy v2.9, Xamarin Android Player v0.6.5, and YouWave Android v5.7. In addition, this study selected four message-hiding applications for experiments and analysis, including Wickr v2.6.4.1, Surespot v65, Cyber Dust v2.6.4, and ChatSecure v14.2.3.

Message-Hiding Applications

In this study, an experiment was performed using four message-hiding applications: Wickr v2.6.4.1, Surespot v65, Cyber Dust v2.6.4, and ChatSecure v14.2.3. Wickr v2.6.4.1 is a free end-to-end message-hiding application that can be used to send text, video, picture, and voice messages. It emphasizes security and anonymity, with no metadata for tracking. Surespot v65 is an end-to-end message-hiding application that provides a symmetric key encryption (256 bit AES-GCM) mechanism, and emphasizes a built-in security mechanism. It can be used to send any message, but only the recipient can read the contents. Cyber Dust v2.6.4 is a message-hiding application that can automatically erase a message without leaving any evidence. All sent messages are deeply encrypted, cannot be accessed again, and cannot be read even by the developer. ChatSecure v14.2.3 is a message-hiding application that provides a powerful encryption mechanism and end-to-end authentication. The encryption methods used include XMPP with TLS for authorization control, OTG for end-to-end authentication, Tor for bypassing firewall restrictions, and SQLCipher for encrypting the locally stored dialogue records.

Description of the Experimental Simulation Environment

This study used the Microsoft Windows 7 operating system as the experimental environment. The system registry, system connection port monitoring, file change monitoring, AVD DDMS (Android Virtual Device Dalvik Debug Monitor Server), and integrated forensics and analysis were used to observe and record changes in files after the Android emulators in this study were installed and run.

Experimental Method Design

The X-Ways Forensics comprehensively analyzed and recorded changes to the local file system and the virtual file system of the emulator. Regshot, Currports, FolderChangesView, and Disk Pulse also recorded and analyzed the local system registry, system connection ports, folders, and files. The experimental and observation results of the Android emulator file system and the message-hiding applications were recorded and analyzed by AVD DDMS and WireShark Android Logcat to find out the names and paths of files that needed to be preserved for forensics. Fig. 1 shows the forensic process and research method.The steps for testing an Android emulator are as follows:

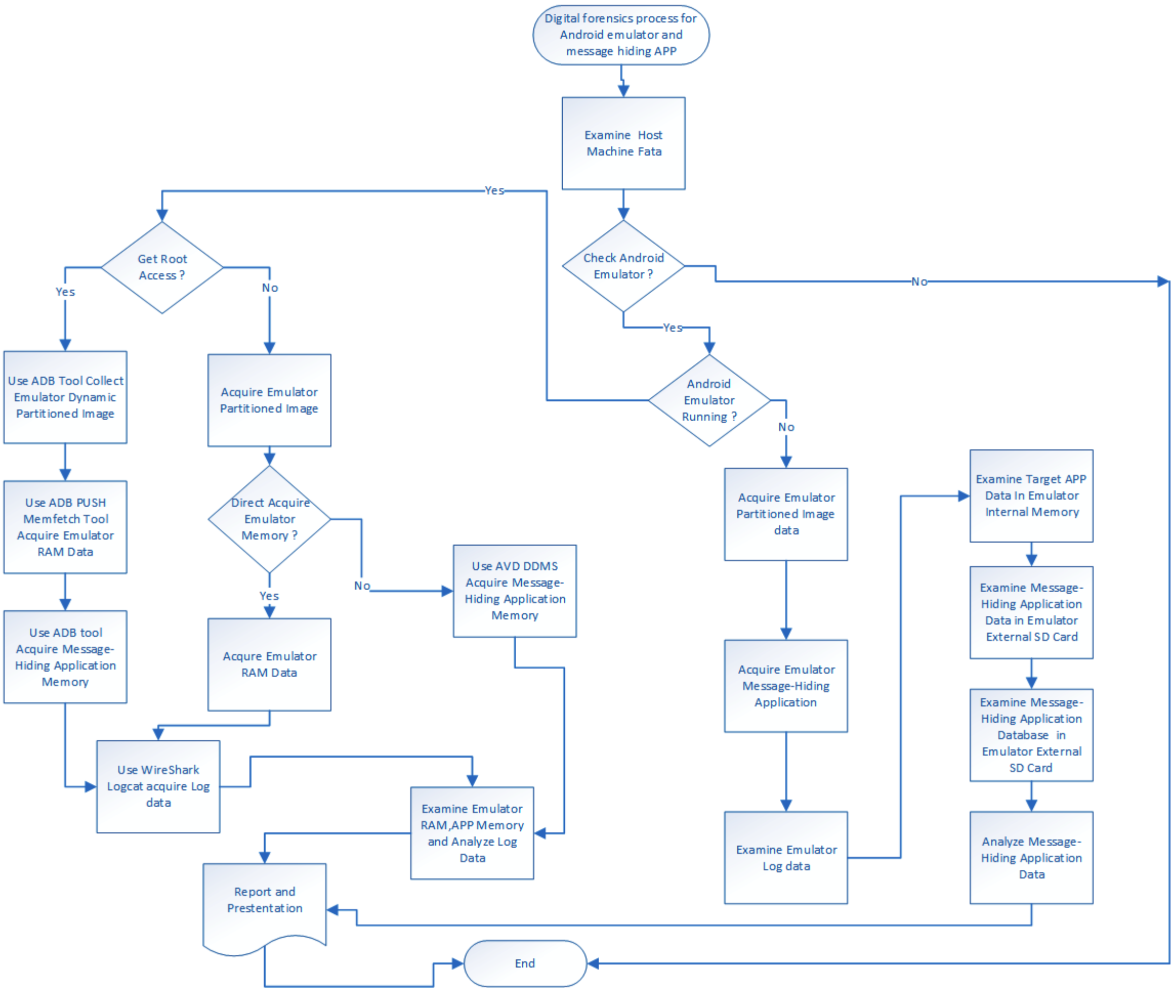
Digital forensics process for Android emulator and message hiding APP
Examine Host Machine Fata
Check Android Emulator ?
No
Yes
Android Emulator Running ?
Yes
No
Get Root Access ?
Yes
No
Use ADB Tool Collect Emulator Dynamic Partitioned Image
Acquire Emulator Partitioned Image
Use ADB PUSH Memfetch Tool Acquire Emulator RAM Data
Direct Acquire Emulator Memory ?
Yes
No
Use AVD DDMS Acquire Message-Hiding Application Memory
Use ADB tool Acquire Message-Hiding Application Memory
Acqure Emulator RAM Data
Use WireShark Logcat acquire Log data
Examine Emulator RAM,APP Memory and Analyze Log Data
Report and Prestentation
End
Acquire Emulator Partitioned Image data
Acquire Emulator Message-Hiding Application
Examine Emulator Log data
Examine Target APP Data In Emulator Internal Memory
Examine Message-Hiding Application Data in Emulator External SD Card
Examine Message-Hiding Application Database in Emulator External SD Card
Analyze Message-Hiding Application Data

Fig. 1 Digital forensics process for Android emulators.

1. Obtain the emulator to be examined forensically, and record its version.
2. Start the Regshot app on the Windows device to take a snapshot of the registry. Then start the Currports, FolderChangesView, and Disk Pulse apps to enable their file monitoring functions.
3. Install the executable file of the Android emulator on the local device and observe the file changes in FolderChangesView and Disk Pulse.
4. After the installation is complete, disable the monitoring functions of FolderChangesView and Disk Pulse, and record the file and folder changes in a report file.
5. Use the Regshot to take a second snapshot of the registry, use the comparison function to analyze the differences in the registry before and after software installation, and generate a report file.
6. Analyze the report files generated in the previous two steps to sort and summarize important file information of the Android emulator, find data items for forensic signatures, and record them.

The message-hiding APPs installed in the Android emulator were tested and observed to determine which files and paths might contain forensic information. During the installation and test of the wireless file transmission and message hiding APPs, X-Ways Forensics was used for comprehensive forensic analysis, and AVD DDMS was used to record and analyze file system changes in the Android emulator.

The steps for testing an APP are as follows:

1. Obtain the wireless file transmission and message-hiding APP to be examined, and record its version.
2. Start the Logcat function of the AVD DDMS on the local device to monitor changes in the file system.
3. Install the APK (Android application package) of the wireless file transmission and message-hiding APP on the local device, observe the file changes using AVD DDMS Logcat, and record the file and folder changes in a report file.
4. Download the heap content (HPROF) of the wireless file transmission and message-hiding APP, respectively, before, during, and after running this APP, and analyze the file system and heap content.

Results and Discussions

Analysis of the installation paths and virtual file systems of Android emulators

Each of the 11 emulators were installed and analyzed. Eight were installed in the locations C:\Program Files\, C:\Program Files (x86)\, or C:\ProgramData\ (three of them can alternatively be installed under a user account). Two were installed under a user account (C:\Users\{USER ACCOUNT}\). One was installed in the root directory of the system disk (C:\KOPLAYER). According to analysis of the virtual file systems of the emulators, we found that seven of the emulators stored the virtual file under a user account (C:\Users\{USER ACCOUNT}\), two of them in the system installation path, and the remaining two in C:\ProgramData\Emulator Program Name.

We analyzed the virtual machine technologies, virtual disk types, and registry keys used by the Android emulators. The emulators in this study used one of three virtual machine technologies: VMware, Oracle VirtualBox, and self-developed LayerCake. They used one of five file configuration formats (virtual disk file types): VMware (VMDK), Oracle VirtualBox (VDI & VMDK), Oracle VirtualBox (VMDK), Oracle VirtualBox (VDI), and self-developed sparsefs. The emulators had different keys in the registry key path HKEY_LOCAL_MACHINE\SOFTWARE\ for forensic personnel to track and examine.

Analysis of the programs started by emulators, ports, and whether ADB Shell can be used

When one of the Android emulators is run, it starts a specific program and port. These allow the emulator to receive and transmit information from and to the host system. The program started by each emulator is stored in the heap while the emulator is running. The test results show that the program started by each emulator might use a different port. By observing the running emulators through the AVD DDMS, we found of the 11 emulators, only KOPLAYER and YouWave Android did not connect to ADB Shell through the respective ports. The remaining nine emulators fetched the dynamically partitioned image file, RAM, and APP heap through ADB Shell.

Analysis of Digital Evidence

This study analyzed types of digital evidence generated by Android emulators. From the analysis results, we found that the file system might contain digital evidence such as files and folders, registry keys, program and network port information, and memory and logs. All of the emulators except KOPLAYER and YouWave

Android fetched the APP heap information. Four of them, Andy, Genymotion, Nox App Player, and Xamarin Android Player, could directly fetch and examine the respective memory locations. The other seven required an importing program and an ADB connection to fetch from memory. The test result showed that the emulators, although using different virtual architectures and virtual environments, allow investigators to obtain APP information using digital forensic procedures and methods. Therefore, digital evidence can still be effectively fetched from these Android emulators. Major forensic signatures for the emulators are summarized in Table 1.

Table 1 Major forensic items for Android emulators.

No.	Emulator Name	Emulator System Path	Path to the Virtual File System of the Emulator	Type of Virtual Disk File	Registry Key Analysis	Analysis on Started Programs and Ports	Whether ADB Shell Can Be Used and Port
1	AMIDuOS	C:\ProgramData\AMI\DuOS\;C:\Users\{USER	C:\ProgramData\AMI\DuOS\imgs	vdi	HKEY_CURRENT_USER\Software\AMI\DuOS\DuOS\;HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Installer\UserData	DuOS.exe:3600; DuoVMHeadless.exe:10088	Yes: 21503
2	Andy	C:\Users\{USER ACCOUNT}\AppData\Roaming\Andy\	C:\Users\{USER ACCOUNT}\AppData\Roaming\Andy\machines\af48496a-085f-4698-8d8a-4d6ce371c7a0(GUID)\images	vmdk	HKEY_CURRENT_USER\Software\Andy\;HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\Andy OS\	AndyConsole.exe:5905	Yes: 5555
3	BlueStacks App Player	C:\Program Files (x86)\BlueStacks	C:\ProgramData\BlueStacks\Android	sparsefs	HKEY_LOCAL_MACHINE\SOFTWARE\BlueStacks\	HD-Frontend.exe:53306	Yes: 5554
4	Droid4X	C:\Program Files (x86)\Droid4X	C:\Program Files (x86)\Droid4X\VirtualBox VMs\droid4x\;	vmdk	HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\Droid4X\	Droid4X.exe:59955	Yes: 26944
5	Genymotion	C:\Users\{USER ACCOUNT}\AppData\Local\Genymobile\Genymotion	C:\Users\{USER ACCOUNT}\AppData\Local\Genymobile\Genymotion\deployed\Mobile device name (ex: Samsung Galaxy Note 3 - 4.3 - API 18 - 1080x1920)	vdi and vmdk	HKEY_CURRENT_USER\Software\Genymobile\Genymotion\;KEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\	Multiple ports such as player.exe:56877	Yes: 5555

No.	Emulator Name	Emulator System Path	Path to the Virtual File System of the Emulator	Type of Virtual Disk File	Registry Key Analysis	Analysis on Started Programs and Ports	Whether ADB Shell Can Be Used and Port
6	KOPLAYER	C:\KOPLAYER	C:\KOPLAYER\deployed\KOPLAYER	vmdk	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\KOPLAYER_is1\	KOPLAYER.exe:537377	No
7	Memu	C:\Program Files\Microvirt	C:\Program Files\Microvirt\MEmu\MemuHyperv VMs\MEmu	vmdk	HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\MEmu\	Multiple ports such as MEmu.exe:57385, 57387, 57391 and MEmuHeadless.exe:21500	Yes: 21503
8	Nox App Player	C:\Program Files\Bignox\BigNoxVM;C:\Users\{user account}\.BigNox	C:\Users\{user account}\AppData\Roaming\Nox\bin\BignoxVMS\nox\	vmdk	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\0147813640F7AF69F569581EE672B6BE1E71798E\	nox_adb.exe:5037, 55504; NoxVMHandle.exe:58001	Yes: 62001
9	Windroy	C:\Program Files (x86)\Windroye;C:\Program Files\WindroyeBox;C:\Users\{user account}\AppData\Local\VirtualStore\Program Files\WindroyeBox	C:\ProgramData\Windroye\vdi;C:\ProgramData\Windroye\Windroye_4E513D9BC016A2AADA0CF6F6426390EB\	vdi	HKEY_LOCAL_MACHINE\SOFTWARE\WindroyeBox\	WindroyeBoxHD.exe:22555; Windroye.exe:55795	Yes: 22515
10	Xamarin Android Player	C:\Program Files\Xamarin Android Player	C:\Users\{user account}\AppData\Roaming\XamarinAndroidPlayer\VMStorageLibrary\Nexus 5 (Lollipop)	vdi	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Installer\UserData\S-1-5-18\Products\21C5AD255AE2DB64E8CB93588A3DFB32\InstallProperties\	AndroidPlayer.exe:49695	Yes: 5555
11	YouWave Android	C:\Program Files (x86)\YouWave Android	C:\Users\{user account}\.Virtualbox\HardDisks	vdi	HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\YouWave\	YouWave Android.exe:60500	No

As shown in Table 2, the Android emulators employ three types of virtual machine software technologies. Digital evidence retained by emulators in the local device includes logs, temporary browser files, registry keys, memory information, files, and folders. Table 2 also shows whether X-Ways Forensics can fetch directly from emulator memory, and the path of the memory file.

Table 2 Digital evidence of Android emulators.

No.	Emulator Name	Virtual Machine Software and Technology Employed	Local Digital Trace Evidence Retained by Emulator	Can Emulator Memory Be Fetched Directly?	Memory Location
1	Andy	VMware (VMDK)	Logs Browser temporary files Registry keys Memory information Files and folders	Yes	C:\Users\MJIB\AppData\Roaming\Andy\machines\06ad203a-81da-46f8-a582-15761de6c68b\images\*.vmm
2	BlueStacks App Player	Self-developed LayerCake virtual technology		No	N/A
3	Genymotion	Oracle VirtualBox (VDI & VMDK)	Logs Browser temporary files Registry keys Memory information Files and folders	Yes	C:\Users\{user account}\AppData\Local\Genymobile\Genymotion\deployed\HTC One - 4.4.4 - API 19 - 1080x1920\Snapshots\*.vmdk
4	Droid4X	Oracle VirtualBox (VMDK)		No	N/A
5	KOPLAYER			No	N/A
6	Memu			No	N/A
7	Nox App Player			Yes	C:\Users\{user account}\AppData\Roaming\Nox\bin\BignoxVMS\nox\Snapshots\*.vmdk
8	AMIDuOS	Oracle VirtualBox (VDI)		No	N/A
9	Windroy			No	N/A
10	Xamarin Android Player			Yes	C:\Users\{user account}\AppData\Roaming\XamarinAndroidPlayer\VMStorageLibrary\Nexus 5 (Lollipop)\Snapshots\*.vmdk
11	YouWave Android			No	N/A

Analysis of the message-hiding APP Wickr

When Wickr is installed, the emulator creates a folder named com.mywickr.wickr2 in the file system, containing its internal memory. This folder consists of five subfolders: app_sfs, databases, files, no_backup, and shared_prefs. The shared_prefs folder stores XML files containing some parameter settings of the APP. The databases folder contains SQLite database files named wickr_db, and log files with the names starting with wickr_db-journal. These two types of files are encrypted, so SQLite database files cannot be opened or browsed by the SQLite database reader, and log files cannot be opened or accessed by common text editors. The files folder stores record files (with the file extension.**wic**) related to the chats with different contacts. All chat record files are encrypted using the SHA-256 algorithm. For each chat record, two .wic files are generated with almost identical names, for example, dacec2704dbdbbff585cdd778a9cb47bbe24a5d583ee2f0d4705bd9e84f1f08f.wic and dacec2704dbdbbff585cdd778a9cb47bbe24a5d583ee2f0d4705bd9e84f1f08f2.wic. The second .wic file is generated empty, for reasons that are not clear. Under the files folder, there is also an encrypted file named keyFile, which is used to store external memory content of the emulator. Analysis of the APP's network connection shows that the APP performs account login and message transmission through the HTTPS (port 443) service, and exchanges messages through the secex.info (204.232.166.114) server.

Analysis of the message-hiding APP Surespot

After Surespot is installed, the emulator creates a folder named com.twofours.surespot in the file system of the internal memory. This folder consists of two subfolders: files and shared_prefs. The shared_prefs folder stores XML files containing some of the APP’s parameter settings. The account information of the last user and the last contact can be found in the surespot_preferences.xml file. The files folder consists of three subfolders: identities, publicKeys, and state. The identities folder stores ssi files of a user account in GZIP format, with the file header 0x1F 8B 08. We used 7-ZIP to decompress the .ssi files, and attempted to parse their content, but we found that the file content was AES encrypted; therefore, we failed to retrieve any account information. Under the publicKeys folder are subfolders storing the account information of the users. The state folder stores the chat records in a file named messages_*user account_contact account*.sss and contact information in a file named friends.sss. All .sss files are stored in GZIP format. After the files were decompressed, we found that the file content was in JSON format, with multiple fields defined, with the following fields and their corresponding values: id, to, hashed, voicePlayed, shareable, iv, fromVersion, gcm, data, from, datetime, mimeType, and toVersion, and with the corresponding values stored. The iv and data fields were encrypted with AES-256. The datetime field contains the chat time stored in UNIX *Numeric-Value* format. The /data/com.twofours.surespot and surespot folders store external memory information of the emulator. Analysis of the APP's network connection information shows that the APP performs account login and message transmission through the HTTPS (port 443) service, and exchanges related messages through the server.surespot.me and appspot.l.google.com servers.

Analysis of the message-hiding APP Cyber Dust

When Cyber Dust is installed, the emulator creates a folder named **com**.radicalapps.cyberdust in the file system of the internal memory. This folder consists of five subfolders: cache, code_cache, databases, files, and shared_prefs. The shared_prefs folder stores XML files containing some parameter settings of the APP. Among these files, the MyPreferences.xml file is of the greatest importance. It contains user account name, device type, and token value encrypted by an private algorithm and encoded in base64. The databases folder stores SQLite database files named cyberdust.db and log files with the names starting with cyberdust.db-journal. The cyberdust.db file permits users to read its content, where the id, message_id, and date fields may help understand certain files of encrypted messages. The files folder contains some useful files for decrypting the messages transmitted by the APP, such as gaClientId, INSTALLATION, privateKey.5715f8afe4b05bbb32a8099f, and publicKey.5715f8afe4b05bbb32a8099f. privateKey.5715f8afe4b05bbb32a8099f is the message ID used in this chat. The gaClientId and INSTALLATION files record UUIDs. No folder is created to store external memory information of the emulator. The analysis of the APP's network connection information shows that it performs account login and message transmission through the HTTPS (port 443) service and exchanges messages through the cyberdustloadbalancerprod-1918061346.us-east-1.elb.amazonaws.com server.

Analysis of the message-hiding APP ChatSecure

After ChatSecure is installed, the emulator creates a folder named info.guardianproject.otr.app.im in the file system of the internal memory. This folder consists of four subfolders: app_KeyStore, databases, files, and shared_prefs. The shared_prefs folder stores XML files containing some parameter settings of the APP. Among these files, the account.xml file is of the greatest importance as it records the user's account information. Under the databases folder, there is a database file imps.db consisting of 21 tables, of which the accounts, contacts, messages, and chats tables contribute the most to forensic examination. The accounts table contains the user's account and plaintext password information. The contacts table contains contact information. The messages and chats tables contain any messages that are not yet deleted. The files folder has an encrypted SQLite 3 database file media.db and an APP debugging track file trail.properties. The track file records metadata that is useful for forensic examination, e.g., the APP start time and database start time. The /data/info.guardianproject.otr.app.im folder stores external memory information of the emulator. The analysis of the APP's network connection information shows that the APP performs account login and message transmission through the XMPP service and exchanges related messages through the jabber.otr.im (port 5222) public server.

As shown in Table 3, the four message-hiding APPs have three types of digital evidence for forensic examination: internal memory file system and external memory of the emulator, network connection analysis data, and emulator and APP heaps. The analysis of the internal memory file system for the emulator shows that Wickr, Cyber Dust, and ChatSecure generate SQLite database files in the program folder to store relevant information. According to analysis of the emulator and APP heap information while each of the APPs are running, secret messages sent and received by the user, and even the already deleted messages, are retained. If the user logs out of an APP account , does not close APP, most of the hidden messages in the APP heap are lost, while related messages can still be found in the emulator memory.

Table 3 Major forensic items for message-hiding APPs.

Process Name	File System of the Internal Memory for the Emulator	Any Folder Generated for the External Memory of the Emulator?	Network Connection Analysis Data	Emulator and APP Heap Analysis
Wickr	com.mywickr.wickr2 folder, Wickr.db, all wic files, and keyFile	No	Account login and message transmission: secex.info(204.232.166.114:443) server; HTTPS service	Secret messages sent and received by the user and deleted messages can be found in the emulator and APP heaps. Most of the secret messages in the APP heap are lost if the user logs out of the APP.
Surespot	com.twofours.surespot folder, user account.ssi, 1.spk, cookie.sss, friends.sss, and messages_user account:contact account.sss	The /data/com.twofours.surespot and surespot folders are generated.	Account login and message transmission: server.surespot.me and appspot.l.google.com(443) servers; HTTPS service	Secret messages sent and received by the user and deleted messages can be found in the emulator and APP heaps. Most of the secret messages in the APP heap are lost if the user logs out of the APP.

Process Name	File System of the Internal Memory for the Emulator	Any Folder Generated for the External Memory of the Emulator?	Network Connection Analysis Data	Emulator and APP Heap Analysis
Cyber Dust	com.radicalapps.cyberdust folder, cyberdust.db, Web Data, privateKey.577aef46e4b07259ae71c8e0, and publicKey.577aef46e4b07259ae71c8e0	No	Account login and message transmission: cyberdustloadbalancerprod-1918061346.us-east-1.elb.amazonaws.com(443); HTTPS service	Secret messages sent and received by the user, and deleted messages, can be found in the emulator and APP heaps. Most of the secret messages in the APP heap are lost if the user logs out of the APP.
ChatSecure	info.guardianproject.otr.app.im folder, imps.db (if the file is not encrypted, plaintext messages may be found), media.db, Web Data, and KeyStore.bks	The /data/info.guardianproject.otr.app.im folder is generated.	Account login and message transmission: jabber.otr.im(5222)	Secret messages sent and received by the user, and deleted messages, can be found in the emulator and APP heaps. Most of the secret messages in the APP heap are lost if the user logs out of the APP,

Conclusion

In this study, 11 Android emulators and four message-hiding APPs were tested to explore digital evidence retained on local devices. Therefore, this study applied existing digital forensic procedures and methods to discover the emulator file structure and file characteristic items in which digital evidence may be hidden on local devices. The results show that investigators can extract digital evidence from these Android emulators when they are used in crimes. The study of message-hiding APPs shows that characteristic items in which digital evidence may be hidden can be discovered based on the internal file system of the emulator, external memory of the emulator, network connection, and emulator and APP heaps. The test results also show that message-hiding APPs with end-to-end encryption have anti-forensic capabilities, posing a major challenge for digital forensic personnel. However, if forensic personnel recover the content of the emulator's internal memory and APPs as soon as possible, they may be able to obtain secret message records that were transmitted. Therefore, the comparison of memory content may help finding favorable forensic items and characteristic items. Message-hiding APPs that provide end-to-end message encryption and database encryption present a major challenge to digital forensic practice. Further exploration is required to study and develop forensic decryption technologies and methods for end-to-end encryption APPs.

References

1. IDC Smartphone OS Market Share 2015, 2014, 2013, and 2012 Retrieved June 30 2016, from http://www.idc.com/prodserv/smartphone-os-market-share.jsp
2. Amiduos Home (2016). Run Android on Windows - Fastest Android Emulator Retrieved May 15 2016, from http://www.amiduos.com/
3. Andy Home (2016). The Best Android Emulator For PC & Mac _ Andy Android Emulator Retrieved May 15 2016, from http://www.andyroid.net/
4. BlueStacks App Player (2016). Bluestacks Android

Emulator for PC and Mac Retrieved May 15 2016, from http://www.bluestacks.com/about-us/app-player.html

5. Droid4X Home (2016). droid4x simulator-best mobile experience on desktop Retrieved May 15 2016, from http://www.droid4x.com/
6. Genymotion Home (2016). Genymotion - Fast And Easy Android Emulation Retrieved May 15 2016, from https://www.genymotion.com/
7. KOPLAYER Home (2016). The Best Free Android Emulator for PC - KOPLAYER Retrieved May 15 2016, from http://www.koplayer.com/
8. Memu Home (2016). MEmu - Android emulator for PC, better than Bluestacks Retrieved May 15 2016, from http://www.memuplay.com/
9. Nox App Player Download (2016). Nox App Player Download for Windows PC, Mac, Laptop Retrieved May 15 2016, from http://noxappplayer.com/
10. Windroy Home (2016). droid4x simulator-best mobile experience on desktop Retrieved May 15 2016, from http://www.droid4x.com/
11. Introducing Xamarin Android Player (2016). Simulate Android apps with the Xamarin Android Player - Xamarin Retrieved May 15 2016, from https://www.xamarin.com/android-player
12. YouWave Home (2016). YouWave, A world for Android on PC Retrieved May 15 2016, from https://youwave.com/