1-1-2015

Network and device forensic analysis of Android social-messaging applications

Daniel Walnycky
University of New Haven

Ibrahim Baggili
University of New Haven

Andrew Marrington
Zayed University

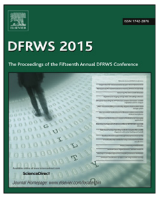
Jason Moore
University of New Haven

Frank Breitinger
University of New Haven

Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin



DFRWS 2015 USA

Network and device forensic analysis of Android social-messaging applications

Daniel Walnycky [a], Ibrahim Baggili [a, *], Andrew Marrington [b], Jason Moore [a], Frank Breitinger [a]

[a] *University of New Haven Cyber Forensics Research and Education Group (UNHcFREG), ECECS Department, Tagliatela College of Engineering, USA*
[b] *Zayed University, Advanced Cyber Forensics Research Laboratory, College of Technological Innovation, United Arab Emirates*

Keywords:
Network forensics
Android forensics
Instant messaging
Privacy of messaging applications
Application security testing
Datapp

A B S T R A C T

In this research we forensically acquire and analyze the device-stored data and network traffic of 20 popular instant messaging applications for Android. We were able to reconstruct some or the entire message content from 16 of the 20 applications tested, which reflects poorly on the security and privacy measures employed by these applications but may be construed positively for evidence collection purposes by digital forensic practitioners. This work shows which features of these instant messaging applications leave evidentiary traces allowing for suspect data to be reconstructed or partially reconstructed, and whether network forensics or device forensics permits the reconstruction of that activity. We show that in most cases we were able to reconstruct or intercept data such as: passwords, screenshots taken by applications, pictures, videos, audio sent, messages sent, sketches, profile pictures and more.

Introduction

Digital evidence from smartphone instant messaging applications is potentially useful in many types of criminal investigation and court proceedings. Text messages have been an important component of the evidence presented in numerous high profile cases in recent years, such as Ashby v. Commonwealth of Australia (Ashby v Commonwealth of Australia (No 4), 2012) and The State v. Oscar Pistorius (S v Oscar Pistorius (CC113/2013), 2014). In the latter, the messages concerned were not sent via Short Message Service (SMS) but by the instant messaging application WhatsApp. Applications like WhatsApp offer users a free or very low cost alternative to SMS for text messaging purposes, and frequently offer other additional features. It is therefore unsurprising that such instant messaging applications have become extremely popular, as a result of which, it is reasonable to expect that more and more cases will involve messages originally sent via such applications.

In this work we perform an experimental forensic study on twenty social-messaging applications for the Android mobile phone operating system. The sum total of the users of the tested applications exceeds 1 billion. Our study illustrates the potential for acquiring digital evidence from the mobile device, data in transit, and data stored on servers.

The remainder of this paper is organized as follows. In the "Related work" section, we discuss related work from the digital forensics and security literature. In the "Methodology" we discuss the research methodology and experimental setup. In the "Experimental results" section we provide an overview of our results, which we discuss in the "Discussion". We propose future work in "Future work" and conclude in "Conclusion".

* Corresponding author.
E-mail address: IBaggili@newhaven.edu (I. Baggili).

http://dx.doi.org/10.1016/j.diin.2015.05.009

Related work

Smartphones are typically kept in close physical proximity to their owners as compared to other potential sources of digital evidence, like computers. This enhances the potential value of digital evidence found on smartphones – suspects may interact with them continuously throughout the day and may take them to the crime scene. In addition to traces of the suspect's communications, a suspect's phone may contain evidence pertaining to their location, and with the advent of the smartphone, they may contain the same rich variety of digital evidence which might be found on computer systems (Lessard & Kessler, 2010). Mobile phones and their applications may be involved in a huge variety of criminal cases, including fraud, theft, money laundering, illicit distribution of copyrighted material or child pornographic images, or even distribution of malware in cybercrime cases (Taylor et al., 2012).

Even before modern smartphones, SMS text messages stored in the GSM SIM card were an important target for forensic examiners (Willassen, 2003). With modern smartphones, mobile applications providing messaging capability may supplement or even supplant SMS, meaning that there may be multiple message repositories on the phone for examiners to retrieve (Husain & Sridhar, 2010).

The majority of new smartphones are shipped with the Android operating system. There have been many different approaches to forensic acquisition of secondary storage of Android devices in the literature since 2009, encompassing both logical and physical acquisition, with some techniques requiring more potential modification to the Android device under examination than others (Barmpatsalou et al., 2013).

Generally speaking, logical acquisition can be performed on an Android device through various backup utilities, and requires no modification of the device or its system software. Physical acquisition techniques described in the literature, on the other hand, often require the installation of a rootkit (modifying the device's system partition) in order to facilitate full access to the device's secondary storage for acquisition through a tool like *dd*, as in Lessard and Kessler (Lessard & Kessler, 2010).

Since these rootkits are often of unknown provenance (in fact, they are most easily sourced from the hacking community), and since any modification to a device under examination ought to be minimized if not outright avoided, Vidas et al. proposed that the Android recovery partition might be more safely overwritten with a known safe forensic boot environment to facilitate physical acquisition of the remaining partitions (Vidas et al., Aug. 2011). By doing so, the system partition is not modified by the rootkit, but full access to the device's secondary storage is obtained by rebooting the device into a modified recovery mode incorporating the necessary software to perform a physical acquisition. This is similar to how a boot CD might be used to facilitate forensic acquisition on a computer system.

From a completeness perspective, physical acquisition is generally preferable to logical acquisition, as a physical image will include any data which exists in unallocated space, such as files that have been deleted but not yet overwritten. Despite this, logical acquisition can still yield significant quantities of digital evidence (Lessard & Kessler, 2010) and is still employed in many studies in the literature (Barmpatsalou et al., 2013; Al Mutawa et al., Aug. 2012; Grover, 2013).

Mobile applications for popular instant messaging or social networking platforms have been the subject of numerous studies in digital forensics literature. Early work on instant messaging applications on smartphones, such as Husain and Sridhar's study on the iPhone (Husain & Sridhar, 2010), examined platforms which were originally released for the Personal Computer (PC) either as a standalone application or via the web. Computer forensic techniques are described in the literature for the examination of artifacts from AOL Instant Messenger (AIM) (Reust, 2006; Dickson, 2006a), Yahoo! Messenger (Dickson, 2006b), other installed instant messaging applications (Dickson, 2006c; Dickson, 2007), web clients for popular instant messaging applications (Kiley et al., 2008), and instant messaging features of social networking websites such as Facebook (Al Mutawa et al., 2011).

As these instant messaging platforms from the PC world migrated to the smartphone with their own mobile applications, so did the digital forensics community move on to investigate activity traces left by these applications on mobile devices (Husain & Sridhar, 2010; Al Mutawa et al., Aug. 2012). In addition to these imports from the PC, instant messaging and social networking applications were developed primarily for the smartphone.

An example of a mobile messaging application is WhatsApp. Anglano analyzed WhatsApp on software-emulated Android devices in recent work providing forensic examiners with information about what data is stored on the Android device by the WhatsApp application, facilitating the reconstruction of contact lists and text conversations (Anglano, 2014). Most of the applications examined in this work fall into the same category as WhatsApp in that they are first and foremost smartphone applications, not PC portovers to Android.

Given the popularity of smartphones, it is not surprising that they have become targets for cyber attacks. Smartphone malware is a growing concern, and the sheer volume of mobile applications brings with it a plethora of potential attack vectors. For example, Damopoulos et al. developed malware which performed DNS poisoning on the iPhone's tethering (also known as personal hotspot) feature, and exposed important user data (such as location and account credentials) when the user employed the Siri service (Damopoulos et al., 2013). In their work on Android inter-application communication and its attendant attack vulnerabilities, Chin et al. found 1414 vulnerabilities in the top 50 paid and top 50 free applications then available for Android on what was then called the Android Market (Chin et al., 2011).

Instant messaging smartphone applications are no exception, as shown by Schrittwieser et al. who examined a set of nine popular instant messaging applications for Android and iPhone, and found vulnerabilities to account hijacking, spoofing, unrequested SMS, enumeration or other attacks on all of them (Schrittwieser et al., 2012). Especially given that smartphones contain so much personal information, it is clear that such threats to their security pose serious risks to user privacy.

While some of these security flaws may assist the digital forensics community to recover more digital evidence from these devices, it is clear that the potential for exploitation of these vulnerabilities by malicious agents will lead to higher incidents of cybercrime targeting mobile devices. Our work complements existing literature by employing network forensics as well as device forensics to provide a more holistic view of what evidence may be obtained from messaging applications on Android devices. Our work also sheds a light on the potential privacy issues that arise from weak security implementations in the tested applications.

Methodology

We selected 20 instant messaging/social messaging applications from the Google Play store based on two factors: keyword results and the number of downloads. The keywords used when searching the Google Play Store were: "chat", "chatting", "date", "dating", "message", and "messaging" to select the 20 applications. Within these search results we wanted to pick a wide range of applications based on a spectrum of popularity. The applications selected range from 500,000 downloads to over 200 million downloads. We would also like to note that we focused on the sections of these applications with one on one communication. For example, we only studied the "Instagram Direct Feature" and not the Instagram feed feature. Another example is that we only studied the direct messages in Snapchat and not "Snapchat Stories".

We performed network forensics to examine the network traffic to and from the device while sending messages and using the various features of these applications. This testing was performed in a controlled lab environment to reduce network variability due to smartphone devices often operating in changing network boundaries. We also performed a forensic examination of the Android device itself to retrieve information from the device pertaining to our activities using each of the applications. Table 5 shows a list of the tested applications in the order they were tested, their version numbers, and the features they support. Video demonstrations of these tests can be viewed at www.youtube.com/unhcfreg.

Network analysis experimental setup

In our research we used an HTC One M8 (Model #: HTC6525LVW, running Android 4.4.2) as well as an iPad 2 (Model #: MC954LL/A, running iOS 7.1.2). We created two accounts for each application using the Android and iPad 2 a week prior to data collection. The Android device was the target of our examination, and the iPad was used simply as a communications partner to exchange messages with the target device. We used a Windows 7 computer with WiFi and an Ethernet connection to the Internet to set up a wireless access point. This PC was used to capture network traffic sent over WiFi to and from both mobile devices. This set up is shown in Fig. 1.

In order to intercept the network traffic, we created a wireless access point to which both mobile devices were connected. This was established using the Windows 7 Virtual WiFi Miniport Adapter feature. This feature allows users to create a virtual network that can act as a wireless access point for multiple devices. To do this, the host computer was connected to the Internet via an Ethernet cable so that the wireless card was not in use. The Ethernet connection was set to share its Internet access with the virtual WiFi Miniport Adapter. We executed the command *netsh wlan set hostednetwork mode = allow ssid = test key = 1234567890* in order to setup the virtual network.

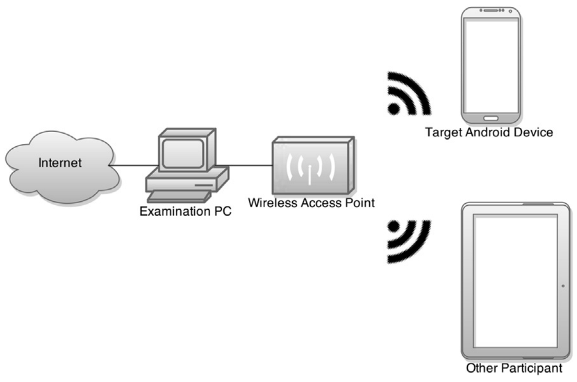


Fig. 1. Experimental network setup.

The network was then enabled using the command *netsh wlan start hostednetwork*. Thus, we were now able to see and connect to the network *test* from our target Android device (the HTC One) and the iPad. Next, we started to sniff the network traffic to and from the mobile devices by capturing data sent over the virtual connection. The number of packets dropped and the capture rate were not recorded, as we did not view it as relevant to the goal of this research. We were focusing solely on the monitoring of real time, low-hanging, unencrypted traffic and whether evidence was captured or not.

Wireshark was used to capture and save the network traffic. These network traffic files (pcap files) can be downloaded from our website (www.unhcfreg.com) under *Data & Tools* upon request. After acquiring the traffic capture files, we examined them with Wireshark, NetworkMiner, and NetWitness Investigator. A full diagram of this setup is shown in Fig. 1. We developed an application to streamline this process, which is highlighted in 6.1.

User activity

Once the network had been setup and the traffic capturing programs were started, we performed a series of actions, which we would subsequently attempt to reconstruct through forensic analysis of the Android device and the captured network traffic. Since every application in our test bank had different capabilities (as listed in Table 5), the actions we performed varied between each application because we wanted to examine all of the messaging types supported. The actions we performed using each application are listed in Table 5.

The content of each message sent with each application was different. We used a pool of evidence types to select from: pictures, videos and plain text words. We pulled one item randomly from each evidence type pool based on each application's capabilities. Messages sent and received were selected from this pool because they deal with the communication aspect, which we advocate should be private, and is a rich source of digital evidence. When a single activity trace was found for an evidence type it was documented. We then proceeded to the next network traffic evidence type on our list of application capabilities in Table 5 until all capabilities were tested.

Data storage experimental setup

After all network analysis was completed, we imaged the phone for data storage analysis. We used Microsystemation's. XRY to perform a logical acquisition of the Android device. XRY is trusted by law enforcement, military, and forensic labs in over 100 countries to assist in digital forensic investigations. We cross validated our results using the free alternative: Helium backup to retrieve application .db files, then using android backup extractor and sqlite database browser to view contents of the .db files. When a single activity trace was found for an evidence type it was documented. We based our storage evidence types on chat logs and clear text user profile data within unencrypted database files. Evidence documented for each application was found in a single .db file that contained the chat logs and/or user information.

Apparatus

Before beginning our examinations, we installed the entire list of instant messaging applications shown in Table 5 on both of the mobile devices. Table 1 shows a list of all software and hardware used during research.

Experimental results

Four out of the twenty applications, namely Snapchat, Tinder, Wickr, and BBM, encrypted their network traffic using HTTPS encryption using SSL certificates. We were not able to reconstruct any data from these applications through traffic analysis, data storage analysis, and server storage analysis due to encryption. The overall results are shown in Table 2.

Packet inspection was not possible with these encrypted applications. However, during our research we found that 16 of the 20 applications tested had unencrypted network traffic and/or unencrypted data storage of some kind. The lack of end-to-end encryption may be due to resources available to developers or because companies deem this data as non-privacy invasive. We posit that organizations that do not expend resources to encrypt their traffic are creating potential security/privacy holes.

No terms of service or security/privacy policies were read before the research. Application pages on the Google Play Store either did not acknowledge security/privacy or reference security/privacy as a key feature. The only exception was Wickr, which emphasized security/privacy.

We would also like to note that false positives occurred when advertisements and profile picture thumbnails were unencrypted, but user content was encrypted. These cases were not documented as evidence. False negatives also occurred when we found activity traces using one tool, but not in another. Therefore, for the traffic reconstruction, we made sure to cross-validate our results using Wireshark, NetworkMiner, and NetWitness Investigator.

Table 1
Devices and tools used for application testing.

Device/Tool	Use	Company	Software/OS version
Laptop	Create Test Network Using Virtual Mini Port Adapter	Windows	Windows 7 SP2
One M8 (UNHcFREGdroid)	Connected to Test Network	HTC	Android 4.4.2
IPad 2 (UNHcFREGapple)	Connected Outside Test Network	Apple	iOS 7.1.2
NetworkMiner	Observe Live Network Traffic	NETRESEC	1.6.1
Wireshark	Observe Live Network Traffic	Wireshark	1.10.8
NetWitness Investigator	Analyze Network Traffic	EMC	9.7.5.9
XRY	Logical Image Creator/Viewer	Micro Systemation	6.10.1
Helium Backup	Create Android Backup	ClockWorkMod	1.1.2.1
Android Backup Extractor	View Android Backup	dragomerlion	2014-06-30
SQLite Database Browser	View Sqlite/DB files	Erpe,tabuleiro,vapour	3.2.0

Table 2
Applications tested with no vulnerabilities found.

Applications	Capabilities	Performed activity	Encrypted network traffic, data storage, and server storage	Emphasized security
Tinder	Text chat	Sent/Received message	Yes	No
Wickr	Text chat	Sent/Received message	Yes	Yes
	Image sharing	Sent/Received image		
Snapchat	Audio, Video and image sharing	Sent image	Yes	No
		Sent video		
		Received video		
BBM	Text chat	Sent/Received message	Yes	No
	Image sharing	Sent/Received image		

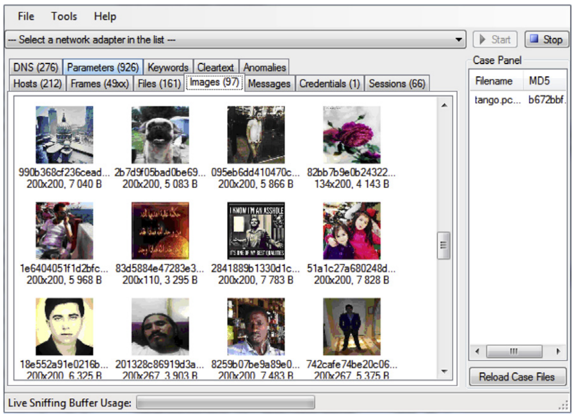

File
Tools
Help
--- Select a network adapter in the list ---
Start
Stop
DNS (276)
Parameters (926)
Keywords
Cleartext
Anomalies
Hosts (212)
Frames (49xx)
Files (161)
Images (97)
Messages
Credentials (1)
Sessions (66)
990b368cf236cead... 200x200, 7 040 B
2b7d9f05bad0be69... 200x200, 5 083 B
095eb6dd410470c... 200x200, 5 866 B
82bb7b9e0b24322... 134x200, 4 143 B
1e6404051f1d2bfc... 200x200, 5 968 B
83d5884e47283e3... 200x110, 3 295 B
2841889b1330d1c... 200x200, 7 783 B
51a1c27a680248d... 200x200, 7 828 B
18e552a91e0216b... 200x200, 6 325 B
201328c86919d3a... 200x267, 3 903 B
8259b07be9a89e0... 200x200, 7 483 B
742cafe74be20c06... 200x267, 5 375 B
Case Panel
Filename
MD5
tango.pc...
b672bbf...
Reload Case Files
Live Sniffing Buffer Usage:

Fig. 2. Captured profile images from Tango.

Captured text messages

There were four instances where we were able to recover the actual text messages that were exchanged between the two devices using network traffic analysis. MessageMe, MeetMe, and ooVoo showed when messages were both sent or received, while we were only able to retrieve text from Okcupid from messages being sent (not received).

Captured multimedia content

We were able to reconstruct different types of media files from the network traffic, including images, videos, locations, sketches and audio.

We reconstructed images when testing Instagram, ooVoo, Tango, Nimbuzz, MessageMe, textPlus, TextMe, Viber, HeyWire, Grindr, and Facebook Messenger. The applications Instagram, ooVoo, Tango, Nimbuzz, MessageMe, textPlus, TextMe, Viber and HeyWire did not encrypt images when they were being received, while Grindr failed to encrypt images when they were being sent.

Some applications include a sketching feature, where one can draw something on the screen and send it to the other party in the communication. We were able to reconstruct sketches from all of the applications listed in Table 5 with sketching features. Furthermore, we were able to reconstruct sketches that were received by our device via Viber and MessageMe, and sketches sent by our device via Kik.

Many of the applications allow users to transmit their location using Google Maps; however, most do not protect this location from being reconstructed by an eavesdropper. We were able to reconstruct location images from TextMe, Viber, and MessageMe when a location was being sent or received. We were also able to reconstruct location images from WhatsApp, Nimbuzz, HeyWire, and Hike only when a location was sent.

As noted in Table 5, numerous messaging applications have the capability to share audio and video between users. Out of all the 10 applications that allow video sharing, we were able to fully reconstruct videos transmitted with Viber, Tango, Nimbuzz, and MessageMe.

Out of all the applications tested that can send/receive audio transmissions, we could only recover the audio files from MessageMe when the audio was being sent from our device.

Beyond multimedia content sent in messages, we were also able to reconstruct all of the images from Tango's newsfeed from the network capture, as well as profile pictures of other Tango users. Interestingly, these profile pictures were not only of contacts we had on our device but seemed to include the profile pictures of other users we could not identify. This is shown in Fig. 2.

Captured URLs for server-side content

During our network traffic testing we combed pcap files using Wireshark to recover links to application servers. It was found that these links were still active and led to the media that was sent/received during our testing. This meant that these applications were storing the media on servers in an unencrypted fashion and with no means of user authentication. This allows for any investigator with access to these links to download the data. All these links were still active weeks after our testing. Table 3 shows some of the URLs found in network traffic from Viber, Instagram, ooVoo, Tango, MessageMe, Grindr, HeyWire, textPlus, and Facebook Messenger, which point to user content stored on servers belonging to those applications.

Chat logs from device storage

As discussed in Section "Network analysis experimental setup", we employed Microsystemation's.XRY to perform a logical acquisition of our target Android device after the

Table 3
Unencrypted user files on application servers.

Application	URL for server-side media
Viber	https://s3.amazonaws.com/share2014-04-21/0d1b42b9f6be43c8b83f0ea4b141f8fae4fb5d775093a17c0e06861c6e2e9300.mp4
Instagram	http://photos-e.ak.instagram.com/hphotos-ak-xaf1/10553994_908375655855764_354550189_n.jpg
ooVoo	http://g-ugc.oovoo.com/nemo-ugc/40051d186b955a77_b.jpg
Tango	http://cget.tango.me/contentserver/download/U8gkjgAAvvzybrAuOcfZPw/9b4IqEqk
MessageMe	http://watercooler.msgme.im/u/1079175176443879424/m/p3joe2.mp4
Grindr	http://cdns.grindr.com/grindr/chat/aa0e6063299350a9b80278feb56a8606acae1267
HeyWire	http://mms.heywire.com/cs/GetImage.aspx?c=p-&p=0%2fmms1%2f20140725%2fp-ec2f6715-afeb-4db4-a5c0-8aaf8ac80689.jpeg
TextPlus	https://d17ogcqyct0vcy.cloudfront.net/377/549/1Kw7ihM5Ri1OkDoXWa.jpg
Facebook Messenger	http://scontent-lga.xx.fbcdn.net/hphotos-xpf1/v/t34.0-12/11156748_10152706456535706_749142264_n.jpg?oh=efbf52c9c9f74525ced5d3d17642ae69&oe=553AE540

Table 4
Unencrypted chat logs in App DB files.

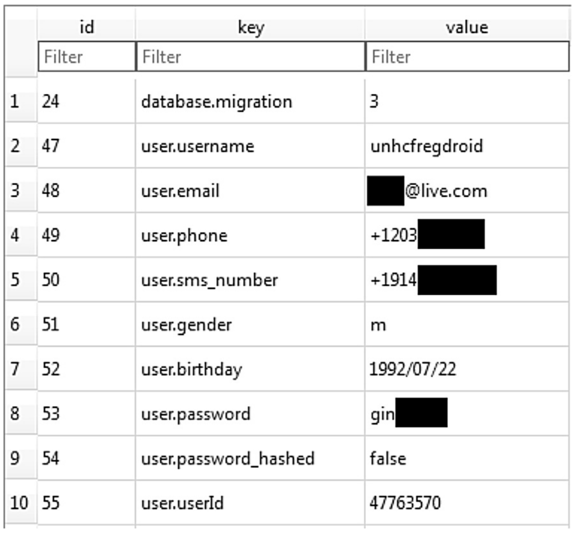
Application	Location of chat logs on android device
textPlus	com.gogii.textplus.ab/textPlus.db
Nimbuzz	com.nimbuzz.ab/Nimbuzz.db
TextMe	com.textmeinc.textme.ab/Database.sql
MeetMe	com.myyearbook.m.ab/Chats.db
Kik	kik.android.ab/kikDatabase.db
ooVoo	com.oovoo.ab/Core.db
HeyWire	com.mediafriends.chime.ab/HWProvider.db
Hike	com.bsb.hike.ab/chats.db

activities described in the "User activity" section. A logical acquisition of a mobile device, much like a logical acquisition of a computer hard drive, misses deleted files and other remnants of data which might otherwise be found in unallocated space. Despite this, we were able to locate the database files that applications use to store data, and from those files we were able to retrieve full chat logs in plaintext of the text messages sent and received by 8 applications (listed in Table 4).

Other user data from device storage

Our examination of the Android device itself revealed some additional user data that could be useful during an investigation. During the examination of the databases that the applications created on the device, we found that TextMe and Nimbuzz stored sensitive user data, including the user's username, email, phone number, birth date, and password in plaintext. Fig. 3 shows user data obtained from TextMe.

Most surprising was a discovery when analyzing the database file for textPlus that the application took and stored screenshots of user activity. When we retrieved these screenshots from the device storage we became concerned that they might be sent from the device to the



	id	key	value
	Filter	Filter	Filter
1	24	database.migration	3
2	47	user.username	unhcfregdroid
3	48	user.email	[redacted]@live.com
4	49	user.phone	+1203[redacted]
5	50	user.sms_number	+1914[redacted]
6	51	user.gender	m
7	52	user.birthday	1992/07/22
8	53	user.password	gin[redacted]
9	54	user.password_hashed	false
10	55	user.userId	47763570

Fig. 3. TextMe's database with user data in plaintext.

application's publisher or to a third-party, but we could find no trace of the images in the captured network traffic. If the screenshots were being sent over the network (and we have no reason to think that they were at the point of writing this paper), then they were first encrypted. We posit that these screenshots could be useful for an investigator seeking to reconstruct user activity on the device.

Summary of results

A summary of all the results of our experiment is shown in Table 5.

Discussion

It is clear from the results shown in Table 5 that the overwhelming majority of the applications in our experiment have significant problems from a user privacy perspective. Users may believe that they are sending messages only to their intended recipient, but in the majority of cases their messages or components of their messages can easily be recovered by network sniffing or by a logical examination of the device itself.

Investigative significance

From a user privacy perspective our results are obviously problematic, but from a lawful surveillance and forensics perspective, it is welcome. With the combination of network traffic analysis and device storage analysis, we created a 360° view of a user's interactions within these applications through an observation of data.

Even though we did not create a new method for analyzing traffic, we contend that our findings are extremely relevant to the forensics community. These findings serve as commentary specifically on the state of messaging applications on the android market today. We aimed to increase user awareness by showing a layer of transparency to the applications tested to eliminate assumptions about security and privacy within them, but also to show investigators how some digital evidence can be easily reconstructed. This "low hanging fruit" that is unencrypted network traffic and unencrypted data storage has proven to be a rich source of digital evidence for potential cases. Protocols used by these applications leave users open to wiretapping because their developers do not offer end-to-end encryption.

Notifying developers

We notified application developers of the vulnerabilities in their software, which permitted us to retrieve so many forensic artifacts. Our experience in attempting this taught us that there is no standardized method to inform application developers of these security issues. We found that there was generally no way to truly notify developers of security issues found from their websites. Even when we sent the companies e-mails through their support contact addresses we rarely ever received a response. This hints towards future research in building a system that can streamline the process of informing companies of security

Table 5
Application capabilities, actions, and traces.

Applications	Capabilities	Performed activity	Network traffic traces	Data storage traces	Server traces
WhatsApp (2.11)	Text chat Audio, video, image, location, and V-card sharing	Sent/Received message Sent/Received voice note Sent/Received image Sent video Sent V-card Sent/Received GPS location	V-Card Location (Sent)		
Viber (4.3.0.712)	Text chat Voice call Audio, video, image, sketch, and location sharing	Sent/Received message Sent/Received sketch Sent/Received image Sent/Received voice note Received voice call Sent/Received GPS location	Images (Received) Sketches (Received) Video (Received) Location (Sent/Received)		Images, Video, Sketches
Instagram (6.3.1)	Text chat Video and image sharing	Sent/Received image	Images (Sent/Received)		Images
Okcupid (3.4.6)	Text chat	Sent/Received message Sent/Received image	Text (Sent)		
ooVoo (2.2.1)	Text chat Voice call Video call Video and image sharing	Sent/Received message Sent/Received image	Text (Sent/Received) Images (Sent/Received)	Chat Log	Images
Tango (3.8.95706)	Text chat Voice call Video call Audio, Video, and image sharing	Sent/Received message Sent/Received image	Images (Sent/Received)		Videos
Kik (7.3.0)	Text chat Video, image, and sketch sharing	Sent/Received message Sent/Received image Sent/Received sketch	Sketches (Sent)	Chat Log	
Nimbuzz (3.1.1)	Text chat Voice call Audio, video, image, and location sharing	Sent/Received message Sent/Received image Sent/Received GPS Location Sent/Received Video	Location (Sent) Images (Sent/Received) Video (Sent)	Plain Text Password Chat Log	
MeetMe (8.6.1)	Text chat Image sharing	Sent/Received message Sent/Received image	Text (Sent/Received)	Chat Log	
MessageMe (1.7.3)	Text chat Audio, video, image, sketch, and location sharing	Sent/Received message Sent/Received image Sent/Received sketch Sent/Received GPS Location Sent/Received Video Sent/Received Music	Location (Sent/Received) Text (Sent/Received) Images (Sent/Received) Sketches (Received) Music (Sent)		Videos
TextMe (2.5.2)	Text chat Voice call Video call Video and image sharing	Sent/Received message Sent/Received image Sent Dropbox file Received video	Location (Sent/Received) Images (Received)	Plain Text Password Chat Log	
Grindr (2.1.1)	Text Chat Image and location sharing	Sent/Received message Sent/Received image	Images (Sent)		Images
HeyWire (4.5.10)	Text chat Voice call Audio, image, and location sharing	Sent/Received message Sent/Received image Sent/Received GPS Location	Location (Sent) Images (Received)	Chat Log	Images
Hike (3.1.0)	Text chat Voice call Audio, video, image, location, and V-Card sharing	Sent/Received message Sent/Received image Sent/Received GPS Location	Location (Sent)	Chat Log	
textPlus (5.9.8)	Text chat Voice call Audio and image sharing	Sent/Received message Sent/Received image	Images (Sent/Received)	App Taken Screenshots, Chat Log	Images
Facebook Messenger (25.0.0.17.14)	Text chat Voice call Audio, video, image, location, and stickers	Sent/Received message Sent/Received image Sent/Received video Sent/Received audio Sent/Received GPS location Sent/Received stickers	Images (Sent/Received) Video Thumbnails (Received)		Images, Video Thumbnails

issues to ensure that decision makers receive these findings. Research like this is important to push both developers and users to care more about security and privacy. A wide range of applications that we tested failed to encrypt their data in one way or another.

Future work

Work still needs to be conducted in this area. For one, these applications change constantly, they receive added features, updated security, etc. An example of this is

Facebook Messenger's new in-app downloads. Applications, whether in Facebook Messenger or not, can store data differently depending on user settings, OS version, and manufacturer. Therefore, continuous testing needs to be performed on new versions of these applications/OSs as they are released to determine what has changed and how much of the prior knowledge of these applications still holds true.

Checking each version of an application for digital forensic artifacts in the manner that this research was conducted would take a long time. This is why the future of testing needs to move towards a more automated process. In our research group, we have been working on a project to analyze the activity of applications installed on a device to determine network traffic encryption and what services the applications are communicating with (see Section "Datapp").

The 20 applications chosen are not the only messaging applications on the Android market, and there is clearly plenty of scope for similar testing to be conducted on other messaging applications. As mentioned before, many of the social media applications, such as Facebook, have their own messenger systems, which also require network traffic analysis. Furthermore, applications that could store and send data securely on one operating system may not do so on another, so testing needs to be performed across different operating systems.

Datapp

There is very little effort required to reconstruct the discussed digital evidence; the tools we used automated the process. Our results can be attained with free tools, and by anyone with a laptop and two small scale digital devices. After we completed our research, we developed our own tool called "Datapp" to further automate the process to make it even easier for anyone to conduct application security and privacy testing. Datapp leverages open source projects like NetworkMiner. The tool automatically creates a wireless network, and sets up packet capture on a system, and displays the results in real-time. This tool was mostly created for the laymen, so that in the future they can test their own applications. Datapp is available for download from our website (www.unhcfreg.com) under Data & Tools.

Conclusion

In this paper, we investigated 20 Android applications through network traffic analysis and server/device storage analysis. This was performed in order to examine the digital evidence that could be of value to forensic examiners and also to evaluate application security in sending/receiving data and application privacy in storing data. Our work showed a variety of results.

We were not able to retrieve any user related information from Snapchat, Tinder, Wickr, or BBM. In the remaining 16 applications, we were able to reconstruct at least some evidence of unencrypted data that was sent and/or retrieved by our device. Despite the advantages for the digital forensics community, these findings are troublesome from a user privacy perspective. It would be quite easy for a nefarious user to sit in a public place with free WiFi, like a café, airport or library, and capture large volumes of personal communications, both in the form of text messages and shared photos. The fact that many of these applications store sent/received media on their servers in an unencrypted format with no user authentication method is also troublesome. It appears that the only user privacy protection in place for such content is the anonymity of the URL.

Overall our work shows that many popular messaging applications have some major vulnerabilities in terms of how they store and transmit data. From a forensics perspective this facilitates the potential reconstruction of large parts of user instant messaging activity, but from a privacy perspective it also exposes the personal communications of users to potentially malevolent third parties.

References

Al Mutawa N, Al Awadhi I, Baggili I, Marrington A. Forensic artifacts of Facebook's instant messaging service. In: Internet Technology and Secured Transactions (ICITST), 2011 International Conference for; 2011. p. 771–6.

Al Mutawa N, Baggili I, Marrington A. Forensic analysis of social networking applications on mobile devices. Digit Investig Aug. 2012; 9:S24–33.

Anglano C. Forensic analysis of WhatsApp messenger on Android smartphones. Digit Investig 2014;11(3):1–13.

Ashby v Commonwealth of Australia (No 4) [2012] *FCA 1411*.

Barmpatsalou K, Damopoulos D, Kambourakis G, Katos V. A critical review of 7 years of Mobile device Forensics. Digit Investig 2013;10(4): 323–49.

Chin E, Felt A, Greenwood K, Wagner D. Analyzing inter-application communication in Android. In: Proc. 9th …; 2011. p. 239–52.

Damopoulos D, Kambourakis G, Anagnostopoulos M, Gritzalis S, Park JH. User privacy and modern mobile services: are they on the same path? Pers Ubiquitous Comput 2013;17:1437–48.

Dickson M. An examination into AOL Instant messenger 5.5 contact identification. Digit Investig 2006a;3(4):227–37.

Dickson M. An examination into Yahoo messenger 7.0 contact identification. Digit Investig 2006b;3(3):159–65.

Dickson M. An examination into MSN messenger 7.5 contact identification. Digit Investig 2006c;3(2):79–83.

Dickson M. An examination into Trillian basic 3.x contact identification. Digit Investig 2007;4(1):36–45.

Grover J. Android forensics: automated data collection and reporting from a mobile device. Digit Investig 2013;10:S12–20.

Husain M, Sridhar R. iForensics: forensic analysis of instant messaging on smart phones. Digit Forensics Cyber Crime 2010;31:9–18.

Kiley M, Dankner S, Rogers M. Forensic analysis of volatile instant messaging. In: Advances in digital forensics IV, vol. 285. Boston: Springer; 2008. p. 129–38.

Lessard J, Kessler GC. Android forensics: simplifying cell phone examinations, *small scale digit*. Device Forensics J 2010;4(1).

Reust J. Case study: AOL instant messenger trace evidence. Digit Investig 2006;vol. 3(4):238–43.

S v Oscar Pistorius (CC113/2013) [2014] ZAGPPHC 793 (12 September 2014).

Schrittwieser S, Frühwirt P, Kieseberg P, Leithner M, Mulazzani M, Huber M, et al. Guess who's texting you? evaluating the security of smartphone messaging applications. In: Proc. 19th Annu. Symp. Netw. Distrib. Syst. Secur.; 2012. p. 9.

Taylor M, Hughes G, Haggerty J, Gresty D, Almond P. Digital evidence from mobile telephone applications. Comput Law Secur Rev 2012;28(3): 335–9.

Vidas T, Zhang C, Christin N. Toward a general collection methodology for Android devices. Digit Investig Aug. 2011;8:S14–24.

Willassen SY. Forensics and the GSM mobile telephone system. Int J Digit Evid 2003;2(1).