Privacy and Data Protection in Smartphone Messengers

Christoph Rottermanner
St. Poelten University of Applied Sciences
Austria
is121023@fhstp.ac.at

Peter Kieseberg
SBA Research
Austria
pkieseberg@sba-research.org

Markus Huber
St. Poelten University of Applied Sciences
Austria
markus.huber@fhstp.ac.at

Martin Schmiedecker
SBA Research
Austria
mschmiedecker@sba-research.org

Sebastian Schrittwieser
Josef Ressel Center for Unified Threat Intelligence on Targeted Attacks, St. Poelten University of Applied Sciences, Austria
sebastian.schrittwieser@fhstp.ac.at

ABSTRACT

Ever since the Snowden revelations regarding mass surveillance, the role of privacy protection in commodity communication software has gained increasing awareness in the general public. Still, during the last years many new messengers were developed for Android, where often privacy was not considered to be a key issue. Due to the widespread use of these apps even in corporate environments this opens up attack vectors that can result in advanced persistent threats. In this paper we analyze the most prominent messenger apps with respect to privacy concepts, focusing not only on the transmission layer regarding the support of encrypted communication, but also attacks targeting the communication metadata, e.g. detecting the existence of communication between users, as well as providing an enumeration of all users of a service. Furthermore, device theft and loss is a major issue regarding the protection of user privacy. Thus, we also analyzed, whether the messages are stored in a secure way on the device itself, or if control over the physical device allows access to the message data. In order to analyze the possible usability of these messengers as means for targeted surveillance of users by the provider (or an entity controlling it), we also analyzed the rights and privileges the respective apps need in order to be able to install and work. Here, major differences could be detected, with several apps claiming privileges that could not be explained with the normal mode of operation, thus posing a serious risk for the privacy of the respective user base.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Mobile Security, Privacy, Messengers, APTs

iiWAS2015, 11-13 December, 2015, Brussels, Belgium.

1. INTRODUCTION & RELATED WORK

Nowadays a lot of people use messaging applications for communication. Therefore, many new messengers were built in the last years with the intention to replace traditional SMS, still most of them are not built with security or privacy in mind. Previous work by Schrittwieser et al. [21] and Mueller et al. [19] already describes various attack vectors and common vulnerabilities against a number of popular messengers. Cheng et al. [10] describe several privacy problems and how easy it was to map a phone number to a user, only utilizing information derived from the messenger application. Furthermore, with the detailed information which is necessary for the registration of some messengers, much sensitive private information was shared with the provider of the application. Another paper from Frosch et al. [14] takes a closer look at the security of the product "TextSecure" and the used protocol. Related work from Unger et al. [24] describes the necessary security features for providing secure messengers. Another work from Mahajan et al. [18] analyses the masses of sensitive information which are processed by messengers like WhatsApp and Viber.

Since Edward Snowden published secret papers from the NSA which contained information about mass surveillance [16, 9], the demand for secure chats increased. Previous work from Stirparo et al. [23] evaluated several other mobile applications with respect to the locally stored private information, the privacy aspects and how much sensitive information was transmitted to the provider, which is especially relevant in a business environment. One requirement for a secure chat is therefore to prevent intelligence services and other third parties from efficient spying on the communication between users, as well as controlling that the provider does not gather private information about the persons who use these messengers. Another aspect lies in checking, how much private information can be extracted from the peo-

ple which were added as friends and what requirements are needed for adding new friends. This is especially important in order to thwart attempts of using the messenger network as means for reconnaissance or early stages of APTs (e.g. phishing attacks).

Altogether, the following contributions are provided in this paper:

- An analysis of the different privacy measures provided by the respective messengers.
- Providing and analysing enumeration attacks for enumerating the user base of the respective messengers.
- Analysing the security of the underlying database environment with respect to privacy protection in case of device loss and theft.

This work is structured as follows: In Section 2 we give an overview on the targeted messenger applications and the methodology while the results are presented in Section 3 regarding privacy aspects and required privileges, Section 4 regarding enumeration of the user base and Section 5 regarding privacy preserving storage of messages on the device. Section 6 concludes the paper.

2. METHODOLOGY

2.1 Selection Criteria for the Apps

The major decision criteria for the selection of messenger apps was that the phone number is required for the registration of a new user and furthermore that the secret code was transmitted via SMS (two-factor authentication). Another criteria for the selection was that some of these applications were already tested in other works in order to be able to compare our results with the results from related work on this topic. A third point was to check new messengers which claimed to support end-to-end encryption, or to put an explicit focus on privacy protection.

2.2 Experimental Setup

The test device for all attacks was a Samsung Galaxy Nexus with Cyanogenmod 11.1.3 (Android 4.2.2). Regarding the required privileges, the newest versions of the apps were downloaded directly from the Google Play Store. With respect to test for user enumeration, we generated a test phone book holding a selection of ten thousand valid entries. In case of the "Line" messenger it was necessary to use the Python library *line* [1]. This library made it possible to perform several brute-force attacks to enumerate users and get further information on the used protocol. Also for the messenger "Telegram" a command line tool [2] was available, which was also useful for the enumeration.

The analysis of the underlying database focussed on attack vectors available through the database interface, i.e. using the database as intended. No file-carving techniques or analysis of internal database structures like shown in [15] have been devised.

[1] `http://carpedm20.github.io/`, Accessed: 2015-05-15
[2] `https://github.com/vysheng/tg`, Accessed: 2015-05-15

2.3 Selected Messenger Applications

In this section we give a brief description of the selected messenger applications. Some have existed for several years, while others were built because of the NSA scandal. All selected applications provide more or less the same functionality.

2.3.1 WhatsApp

This messenger is the most popular one in the Android environment. It is free for one year and available for multiple platforms. A current download statistic from Google Play store showed that the application has been installed between 1.000.000.000 and 5.000.000.000 times (status from 22.04.2015). The main features provided by this messenger are sending messages to other people, creation of group chats and sharing media like pictures and videos. Furthermore, there is no need to add contacts to this application, because the application looks up for registered WhatsApp users in the address book.

2.3.2 Line

Another popular messenger is Line from Japan. In 2014 it had around 600 million users and analysts expect that it will reach 700 million users in 2015 [13]. The download statistic from Google Play store shows, that this messenger has been installed between 100.000.000 and 500.000.000 times (status from 22.04.2015). This application has the ability to send messages, perform audio/video calls and share the current GPS-coordinates. It can be used on smartphones, tablets and personal computers. Like WhatsApp it is possible to create group chats or chats with a single person. Since v4.5.3 of the Android application, the support for end-to-end encryption was added including the ability to destroy messages after a specified time [22].

2.3.3 WeChat

Like Line, this messenger also supports messaging, audio/video calls, large group chats, as well as sharing of media files and the current GPS position. Furthermore, this application provides the feature to communicate just by shaking ones phone. After this, contacts which also shake their phone at the same time, are listed in the application. The current download count from the Google Play store shows, that this messenger has been installed between 100.000.000 and 500.000.000 times (status from 22.04.2015).

2.3.4 Telegram

This new messenger was developed after the revelations by Edward Snowden in order to provide the ability to communicate securely. With this messenger, group chats and communication with a single person are possible, as well as file sharing. The secure chat provides a self-destruction timer which deletes all messages older than a defined age. However, this secret chat is not available for groups at the moment. For the encryption [4] of the communication between the server and the client, a custom developed encryption algorithm is in use.

Current numbers from the Google Play store show that this messenger has been installed between 50.000.000 and 100.000.000 times (status from 22.04.2015).

2.3.5 TextSecure

This messenger advertises with the usage of strong encryption while being easy to use for non-experts and as reliable as making normal phone calls or sending normal text messages. Every message, group chats as well as private chats, is encrypted using end-to-end encryption. Furthermore, the provider has no access to metadata for group chats like group members, group title or the group avatar icon [5]. In Version 2.7.0, the support for SMS and MMS [6] encryption ended due to security flaws of SMS/MMS and other problems with the key exchange. Current numbers from the Google Play store show that this messenger has been installed between 500.000 and 1.000.000 times (status from 22.04.2015) [5].

3. REQUIRED PRIVILEGES AND PRIVACY ASPECTS

Since Edward Snowden published papers about the NSA-surveillance, awareness regarding privacy increased. This section describes the different problems and implemented privacy features for the selected messengers. This includes checking, which data is shared with the providers of the respective messengers and whether it is possible to define, which person can see private information about a user. Another part of this research is to determine, if the applications only use the permissions from the phone when the user is using a feature of the messenger where these permissions are necessary. Finally, this section also covers a review on the privacy relevant permissions needed in order to install the selected messengers.

3.1 Line

The installation of this messenger requires several permissions providing the application with access to a large amount of information on the user of the phone. However, it was discovered that the application only requests these permissions, if the user utilized functions which require them. The following list shows the main permissions, which are required to install Line, solely listing the permissions that could leak private information about the user:

- **Device & App history:** Retrieve running apps.
- **Identity:** Find accounts on the device.
- **Contacts:** Read your contacts.
- **Location:** Approximate location (network-based), Precise location (GPS and network-based), Access extra location provider commands.
- **SMS:** Receive text messages (SMS).
- **Phone:** Directly call phone numbers, Read call log.
- **Photos/Media/Files:** Read the contents of your USB storage, Modify or delete the contents of your USB storage.
- **Camera:** Take pictures and videos.
- **Microphone:** Record audio.
- **Device ID & Call information:** Read phone status and identity.

With this large amount of permissions, it would be easy to use this application for tracking and spying on the user. But as already mentioned, it could not be discovered that the application accessed permissions while the user did not use a feature which required this. Still, for exact information about the usage of the permissions, it would be necessary to track these for a long time.

During the user enumeration (see Section 4.1), the requested user received a message, asking whether the user should be blocked or added. If the target pressed *add*, it was possible for the attacker to see the posts of the victim. Furthermore, when blocking the request, it was still possible to see the photo and the current status message. Thereby it is often easy to collect a photo of the specified user only requiring the user ID, QR-Code ID or the phone number of the person.

A feature of this messenger is the option to decline the upload of the personal phone book during the authentication. Also after that phase it is possible to configure, whether the application should upload contacts or not. Another feature of Lines is the so-called *private chat*. This version of the chat offers end-to-end encryption for transmitting personal information. Still, this feature is not available for group chats and it is necessary to explicitly activate the private chat, thus allowing the provider to see all communication in group and non-private chats.

Another feature provided by the Line messenger is the possibility to identify a user solely using a user ID. Thus, no phone number must be shared with contacts or maybe even unknown people, allowing to hide this (sometimes personal) information. Another privacy related feature is that only the user token is transmitted in a message and no contact information can be retrieved from the message content, again preventing the publishing of the user's phone number (especially considering using the user ID as means for setting up communication).

When adding a new contact it is possible to see all the posts of this user, the only thing required is an acceptance of the so-called *friendship*. Most of the users which were added during the enumeration experiment (see 4.1), willingly accepted the new contact, hence it was possible to see the posts of these users. Lacking the willingness to challenge friend requests is a major problem which cannot be solved on a technical basis, but relates to a lack of awareness regarding privacy issues in the broader public. This especially holds true since Line allows the user to define groups, where posts are only exposed to members (i.e. contacts) belonging to the respective group.

Following we analyze data that is leaked during the normal operation of the messenger app:

During the authentication process, one of the first protocol steps with the server is the HTTP request shown in Listing 1.

Listing 1: Request during the verification process for the messenger Line

```
GET /API/androidevent.php?oursecret=
    line10042014jo&udid=
    adxid9bc46bkeplki5imt&androidID=&
    macAddress=&type=&storeAppID=&
    device_name=Galaxy%20Nexus&device_type=
    android&os_version=4.2.2&country_code=
    US&language=en&app_id=jp.naver.line.
    android&fbattribution=null&event=
    LINE_Waiting_For_SMS&data=&uagent=&
    currency=&custom_data=&idfa=9c361039-9
    b75-47bb-9f52-ea3a589d058e&isLAT=false
```

```
HTTP/1.1
Host: apps.ad-x.co.uk
Connection: Keep-Alive

HTTP/1.1 200 OK
Server: nginx/1.0.15
Date: Mon, 30 Mar 2015 18:09:53 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/5.3.3

17
<Success>true</Success>d
0
```

The major privacy related issue in this message is that the exact version of the used phone is transmitted to the server, which seems unnecessary for the authentication process, whereas this information can help to identify the device in question for further tracking. In addition, the verification process also contains information about the used phone, as outlined by Listing 2.

Listing 2: Sensitive information within request during the verification

```
.....x...Q.t.K..P..'|.R-"ZF..k
    ..Z
n..H[...-_.H?.......y...L␣.y...n]E..U....N5
    .....O#.....B..4␣....{.1a..|.G.q!...$
    ..UU^.k.Z(^N.ZW\/~8.Fz
..A.....
U..!:..#.>F.?.:X...d.,...../2..lG..F4.MU.!H
    /g.UQ.....@}.............!..
    startVerification(.AT...
+43676XXXXXXX.␣
    d02247b8d34b694f80d7d6fbff648395...
    Galaxy␣Nexus.
Android␣OS..4.2.2..Galaxy␣Nexuse..␣␣<————
NOT_SPECIFIED.@...23203(.en_US.
```

3.2 Textsecure

One of the aims of TextSecure lies in trying not to save any metadata of the users. The telephone numbers of the contacts are uploaded hashed, as mentioned in Section 4.2. The problem with this solution is that the range of combinations is very small. Furthermore, this allows generating all possible hashes and comparing them with the uploaded hashed contacts of a user. As the developer of TextSecure already mentioned, there is no practicable solution available to hash the contacts securely. Several concepts are already in development, but unusable for a large amount of users [1].

Listing 3 shows how the telephone numbers look like when they were uploaded to the server. A drawback regarding privacy is the fact that the user does not have the option to decline the upload of the phone book in TextSecure, hence, all the contacts from the local phone book are uploaded.

Listing 3: Uploading hashed contacts for the TextSecure messenger

```
https://54.172.208.191/v1/directory/tokens
{
    "contacts": [
        "hr/5JNlZd7AgnQ",
        "lLkSRf60EHM8tA",
        "BprFLzDEJZnJyw",
        "+k6SXgmv1mCQJw",
        "Lroio4/R1J6H9g",
        "4Hq67sQhRD0CQQ",
        "t7gsZUtUI19l2w",
        "ez5uE0BDpET8yQ",
        "R8SwrCsuRrlcfA",
        ....
```

Nevertheless, the provider gains no knowledge on the user's plaintext contacts, however, a crucial error with respect to privacy is the fact that the phone number of the destination is stored in plaintext within every message (see the example in Listing 4, for privacy reasons the last digits of the phone number were changed).

Listing 4: Sensitive information within every message

```
https://52.1.136.37/v1/messages/+43650
    XXXXXXX
[..]
{
    "destination": "+43650XXXXXXX",
    "messages": [
        {
            "body": [..],
            "destinationDeviceId": 1,
            "destinationRegistrationId":
                8217,
            "type": 3
        }
    ],
    "relay": null,
    "timestamp": 1429638343114
}
```

The same behavior could be determined when receiving new messages: The data structure even possesses a data field, which contains the phone number of the source in plaintext (Listing 5). This means that while the provider does not have access to the actual messages, he is able to locate all communication paths via this metadata, thus allowing to generate a detailed overview on when and with whom users communicated.

Listing 5: Sensitive information within every message

```
https://52.1.136.37/v1/messages/
[..]
{
    "messages": [
        {
            "message": "MwohBR816l3tB+
                AhuO1zdTUqhJTB␣<...>",
            "relay": "",
            "source": "+43650XXXXXXX",
            "sourceDevice": 1,
            "timestamp": 1431696170231,
            "type": 1
        }
    ]
```

```
}
```

Furthermore, the authentication token (see Listing 6) for the TextSecure contains the telephone number of the current user.

Listing 6: Encoded authentication token of the messenger TextSecure

```
Authorization: Basic
    KzQzNjY0WFhYWFhYWDpMZXJjbTBLRFZTS0tk
ZklCNGZIRXcrbnU=
```

That results in the decoded string shown in Listing 7.

Listing 7: Decoded authentication token of the messenger TextSecure

```
+436642356829:Lercm0KDVSKKdfIB4fHEw+nu
```

Another feature regarding privacy that is provided by TextSecure is that all messages of the group, as well as private chats are encrypted by default. The provider cannot intercept any messages of the communication between the users, even one of the documents leaked by Edward Snowden indicates that at that time even the NSA was not capable to decrypt OTR [2] communication, which is partly used in the TextSecure protocol. Furthermore, related work from Frosch et al. [14] confirms that currently the protocol is secure against known attacks and that the found vulnerabilities were fixed by the developers.

Also the circumstance that the application only requires a few permissions to work properly is beneficial with respect to privacy. The following list shows the main permissions, which are required to install TextSecure, solely listing the permissions that could leak private information about the user:

- **Identity:** Read your own contact card, Modify your own contact card, Find accounts on the device.
- **Contacts:** Read your contacts, Modify your contacts.
- **SMS:** Read your text messages (SMS or MMS), Edit your text messages (SMS or MMS).
- **Phone:** Read call log.
- **Photos/Media/Files:** Modify or delete the contents of your USB storage, Read the contents of your USB storage.
- **Device ID & call information:** Read phone status and identity.
- **Other:** Full network access.

Overall, TextSecure only requires 24 permissions to work properly. As the list shows, there is no way to track the user via GPS, neither the messenger requires the privilege to enable the microphone to listen. Furthermore, it is no problem to use another application on the phone to record some music, take a picture with the camera and send this media file via TextSecure to the other users.

To guarantee, that the chat partner is really the person that he/she claims to be, it is possible to compare a fingerprint of the key, which is used to encrypt the messages. To perform this, meeting the other user in person and scanning the generated QR-code of the key is necessary. If the key of the chat partner changes, the user receives a message that the key changed and whether the user wants to accept the new key. Therefore, an attacker cannot easily intercept the communication and provide a fake key, which is then accepted by the application without confirmation of the user.

Another feature is that the user can define, whether it is possible to make a screenshot of the on-screen conversation. This feature does not prevent the chat partner from making screenshots, but it prevents that another application on the phone can make a screenshot. Furthermore, TextSecure provides a feature to set a password that encrypts the local messages and the encryption keys used. Thus, all the sensitive local data can be protected and even if the phone is stolen, the private messages should remain safe (see 5.1). Furthermore, the user can define a validity period for the password, i.e. the application "forgets" the password after a certain time forcing the user to reenter it. This is an additional protection mechanisms in case the phone is lost or stolen.

3.3 WhatsApp

For the WhatsApp messenger, it is not possible to decline the upload of all the contacts from the phone book, hence, the server retrieves information on all contacts from ones personal environment. Another problem is related to the feature controlling who can see the last time a user was online [25]. In the settings there is an option that allows to define that nobody can see what the last read message was, nor when the user read a specific message. Therefore, no one should see when the user was online for the last time. The problem with this setting is that everybody can see the current online status of a user. This means that tracking of the online status is still possible when this is performed continuously every second. For this tracking the tool WhatsSpy [3] was developed. Using this tool it is possible to track the online activity of every WhatsApp user around the world by continuous monitoring of the current online status. Furthermore, the user has no capability to change that setting, thus currently possessing no countermeasure against this form of tracking. As part of this work, the tool WhatsSpy was installed on a Raspberry Pi and twelve users were monitored, to check for the detail of information that can be gathered during the surveillance. Using this information it is possible to e.g. determine the time a user wakes up in the morning, thus allowing detailed analysis of user behaviour. Hence, the information can be used to generate a timeline for the user, Figure 1 gives an overview on the reconstruction detail.

For some users, we were able to reconstruct daily routines using this type of information, e.g. it could be determined when they go to work and come home. Moreover, with this tool it is possible to determine, whether users are using the same group chats and therefore check, which users may be connected. Within just one week of surveillance it was possible to gather enough information about the daily routine of some users and this information could be more detailed when enriched with other external information together with a longer surveillance period. Unfortunately, everybody around the world has access to this information and the user has no option to change that.

On a side note, additional information is leaked through the profile pictures of the users and their status messages,

[3] `https://gitlab.maikel.pro/maikeldus/WhatsSpy-Public/wikis/home`, Accessed: 2015-05-27

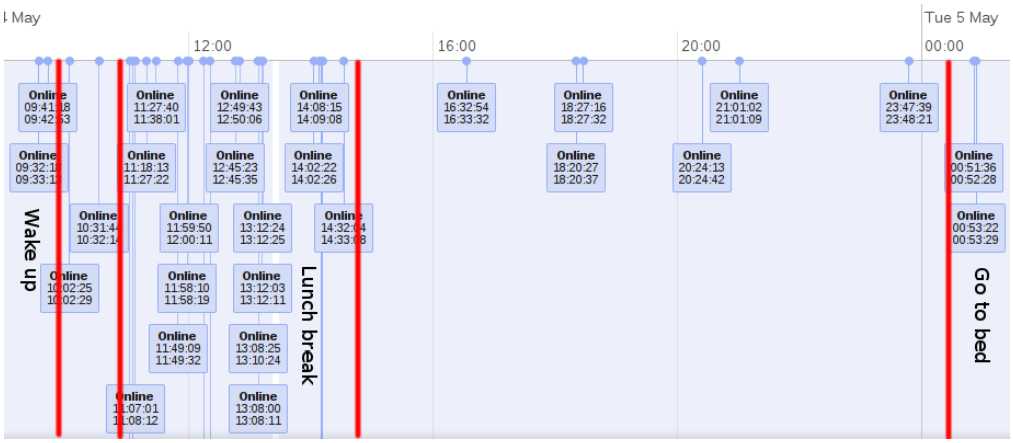

Figure 1: Timeline from WhatsSpy of a single user

which in many cases contain private information. Here, WhatsApp provides the feature allowing the user to define which other users can see the current status message and the profile photo. In opposite to the settings for the online activity, these features are working as intended. A problem with that setting is that the default is set to full public access. During testing WhatsSpy, only one of the twelve monitored users changed this default setting.

As related work [8] states, WhatsApp tries to implement the end-to-end encryption protocol from TextSecure. Still, a recently posted article [8] shows that this feature is not usable for all WhatsApp users and that nobody can confirm, whether they are actually using the new end-to-end encrypted chat. Furthermore, the messenger does not provide a secure chat like the messengers Line, Telegram and TextSecure. Therefore, the users do not have the assurance that the provider cannot monitor and analyse all traffic. The following list shows the main permissions, which are required to install and operate WhatsApp, solely listing permissions that could leak private information about the user:

- **Identity:** Add or remove accounts, Find accounts on the device, Read your own contact card.
- **Contacts:** Modify your contacts, Read your contacts.
- **Location:** Approximate location (network-based), Precise location (GPS and network-based).
- **Photos/Media/Files:** Read the contents of your USB storage, Modify or delete the contents of your USB storage.
- **Camera:** Take pictures and videos.
- **Microphone:** Record audio.
- **Device ID & call information:** Read phone status and identity.
- **Other:** Read sync statistics, Pair with Bluetooth devices, Change your audio settings, Modify system settings, Use accounts on the device, Full network access, Read Google service configuration.

Many of these permissions can be abused by the provider, in order to monitor sensitive and private information about the user. Overall, this application needs thirty-five permissions to work correctly.

3.4 Telegram

Like the messenger Line, Telegram offers a secret chat. This kind of chat is end-to-end encrypted and cannot be read by the provider. It also provides a mechanism for perfect forward secret [3]. If the user has sent more than one hundred messages or has used the key for longer than one week, a new key is generated and all the old messages cannot be decrypted anymore. Even with the new key, it is not possible to reconstruct any old keys, disallowing the possibility to reconstruct old messages. However, the messenger keeps a local unencrypted copy of old messages, which completely thwarts any attempt at providing forward secrecy in the case of stolen or lost devices. Another feature of the secret chat is a self destruction timer quite similar to TextSecure. Furthermore, it is possibility to see whenever a user took a screenshot of the on-screen conversation, still, currently the secret chat is not set as default mode of operation and needs to be initiated explicitly by the user. Currently, it is also not available for group chats.

Similar to WhatsApp, Telegram also provides the feature to change the visibility of the online status without introducing the same problems. Thus, it is not possible to see when the user was online for the last time and if the user is currently online. Another feature is the ability to set a password for the application as user, requiring to unlock the messenger to read new messages. In case the password is lost, all secret chats are lost and the application needs to be reinstalled. The application also provides an *account self-destruct* feature, deleting all chats and information on the user after a preset period of inactivity.
The following list shows the main permissions, which are required for Telegram, solely listing permissions that could leak private information on the user:

- **Identity:** Add or remove accounts, Find accounts on the device, Read your own contact card.
- **Contacts:** Modify your contacts, Read your contacts.
- **Location:** Approximate location (network-based), Precise location (GPS and network-based).
- **Photos/Media/Files:** Modify or delete the contents of your USB storage.
- **Camera:** Take pictures and videos.
- **Microphone:** Record audio.

- **Device ID & call information:** Read phone status and identity.
- **Other:** Create accounts and set passwords, Read Google service configuration, Draw over other apps.

Like TextSecure, this messenger also needs only few permissions to work properly, still requiring permissions like camera, microphone or GPS that could be abused for user tracking.

3.5 WeChat

The WeChat messenger is quite similar to Line and thus provides similar features regarding privacy protection. One setting controls, whether a user can be found by the phone number and if friendship requests are accepted automatically. Hence, the user has the choice to prevent being found by other users. This feature is also available for the WeChat-ID, but not for the QR-code (see Section 4.5). Furthermore it is possible to define whether the phone book should be uploaded to the server and the user has to explicitly confirm this during the upload procedure.

Another privacy related feature is that the users can define, if they want to share the posted messages (so called *Moments*). Furthermore, it is possible to make the *Moments* public or only available for friends of the user. WeChat does not provide a secure chat, therefore all communication is unencrypted and can be monitored by the provider. Furthermore, the provider of the messenger is allowed to track a great amount of information while the user is using the voice feature [7]: This information includes name, telephone number, email address, credit card details, profile biography and profile picture, which has given rise to suspicions of governmental monitoring of persons [12].

As the list of permissions below shows, this application could be easily used to monitor the users and gather private information about them. Only the most critical permissions for gathering private information about the users are listed. Furthermore, the provider is able to use these permissions to track the current location, take pictures from the places and use the microphone to record sounds within range of the phone. However, is was not possible to detect any misuse of these permissions during our analysis

- **Identity:** Add or remove accounts, Find accounts on the device, Read your own contact card.
- **Contacts:** Modify your contacts, Read your contacts.
- **Location:** Approximate location (network-based), Precise location (GPS and network-based).
- **Photos/Media/Files:** Read the contents of your USB storage, Modify or delete the contents of your USB storage.
- **Camera:** Take pictures and videos.
- **Microphone:** Record audio.
- **Device ID & call information:** Read phone status and identity.
- **Other:** Download files without notification, Read sync settings, Create accounts and set passwords, Modify system settings, Measure app storage space, Access Bluetooth settings, Pair with Bluetooth devices, Change your audio settings, Draw over other apps, Use accounts on the device, Full network access.

4. USER ENUMERATION

Most of the applications upload the phone book of the users to a server in order to check which of the contacts are also registered for the respective application. As an attacker, it could be interesting to generate a list of all users currently subscribed to a given messenger, e.g. in order to launch impersonation attacks [21]. Furthermore, in many cases it is possible to extract additional information on the users like availability, status messages, pictures and other metadata that can be used in order to launch phishing attacks. Having access to the phone book of a person is also a very powerful starting point for analyzing the social network of said user during the reconnaissance phase of an targeted attack. This enumeration attack is typically carried out by uploading all possible telephone numbers from a certain number range (e.g. a country or a region) using a valid account and check for numbers known by the service [19]. In order to check for this vulnerability, we generated a test phone book holding a selection of ten thousand valid entries. Furthermore, it was checked, whether the messenger throws error messages or aborts the communication during the synchronization process. For the messengers Line, Telegram and Textsecure, it was possible to develop short scripts to efficiently enumerate large amounts of users.

4.1 Line

For the messenger Line, three methods for the enumeration of users exist: (i) Upload of phone numbers, (ii) upload of User IDs and (iii) the scanning of special QR-codes.

Due to the unknown protocol structure, it was not possible to create an automatic tool to upload the phone numbers directly to the server, instead we used the synchronization process: The phone numbers were added to the contact storage and on start-up, the messenger synchronized all the contacts from this storage with the messenger server. This synchronization process has no limit regarding the number of contacts.

Another way to enumerate the users lies in brute forcing the user ID with a Python library [4]. Within approximately two hours it was possible to enumerate around three thousand valid accounts. Again, no limitation exists, allowing to enumerate all users, which have the *user ID* feature activated.

The last method for the enumeration used the QR-code feature. These codes contain HTTP-links and use a ten character long ID containing only letters (upper and lower case), as well as numbers to identify a user, e.g.: *http://line.me/ti/p/SQXhW*

During the enumeration, the connections time out after the first thousand requests, still, it is possible to simply reconnect. Furthermore, a distributed approach using several clients could be implemented.

4.2 TextSecure

For the TextSecure application, it is possible to save a large amount of contacts directly in the local phone storage. Based on these contact details, all telephone numbers are uploaded to the server, in order to check if these numbers are registered TextSecure users. This procedure can also be automated with a script with the only additional re-

[4] `http://carpedm20.github.io/`, Accessed: 2015-05-15

quirement that the numbers need to be hashed before they are uploaded to the server. Listing 8 illustrates how this is performed.

Listing 8: Hashing of the phone number for TextSecure

```
import hashlib
import base64

phone_number = '+436501234567'

digest = hashlib.sha1() # load function to
    hash with sha1
digest.update(phone_number.encode()) # Set
     number to hash
hashed = digest.digest() # Hash the number
hashed = base64.b64encode(hashed) #
    Perform base64 encoding
print(hashed[0:14].decode()) # Use the
    first 14 characters of the result
```

For validation, this generated information is uploaded to the server using the structure provided in Listing 9.

Listing 9: Structure of the uploaded phone numbers

```
{
    "contacts": [
        "yw9jWjOGL6QbAQ",
        "qPQWViTIcn+0qA"
    ]
}
```

With this information it is possible to perform an automated enumeration, however, there can be instances where a slightly different algorithm is used for generating the hashes: Sometimes the last character of the hash is not calculated correctly, compared to the results of the Android application. Hence, all possible last characters were inserted into the structure as a workaround.

Another way to enumerate the users of TextSecure is to check, whether there is a public key available for the requested user ID. This key is required for further communication with the specific user and therefore publicly available. It was possible to enumerate around six hundred users from Austria within two days using this technique. The problem is that currently not many users are registered for this messenger and therefore many possible user IDs are not active.

4.3 WhatsApp

A similar approach works for WhatsApp: The generated phone book is directly added to the contacts, all these entries are uploaded to the WhatsApp server to check if the contacts exist and the answer is returned to the local device. There is no limitation, which allows to enumerate a large amount of users within a short period. Figure 2 shows a part of the results given by the enumeration. For privacy reasons the faces of the users were censored.

4.4 Telegram

The same vulnerability also exists for the messenger Telegram. With an open source command line tool for Linux, it is possible to enumerate a large number of accounts. Still, it was not possible to use the same procedure as for the other

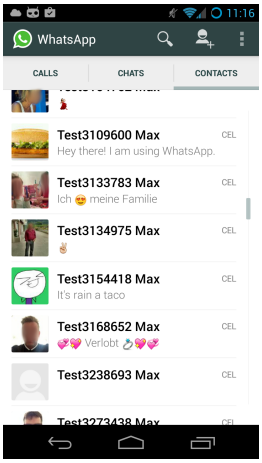

Figure 2: Enumerated users by their phone number with WhatsApp messenger

messengers: It seems that the messenger does not check all the contacts, whether they are using the Telegram messenger at once. This could be a kind of limitation, in order to prevent enumeration attacks, still using the command line tool and a Python script, it was very easy to enumerate available users. This circumstance suggests that the check against this kind of attacks is implemented on the client side. However, it could not be confirmed, that the messenger application actually enforces a limitation, because a custom protocol [4] is used for the communication, which was not analysed in the course of this work.

4.5 WeChat

As for the other messengers, the flaw of user enumeration could also be determined for the application WeChat. Like the messenger Line, the same methods for adding friends are available.

Again, it is possible to enumerate users by uploading a large amount of random phone numbers, combined with the feature to search a contact by its user ID. The feature to add a new contact via the user ID also allows a wildcard search, which makes it possible to find a large amount of contacts by using one character in the search box. It is also very easy to define a word list, however, for an automated attack, it would be necessary to reverse engineer the used protocol in order to determine the correct structure for the enumeration. Unlike the two other methods, the QR-Code search was not practically for a fast enumeration of the user base, because the range of possible QR-Codes includes all letters, numbers, as well as special characters. Furthermore, the string used for user identification in the QR-Code possesses a length of twenty characters, e.g. $http://weixin.qq.com/r/m6t6YszEiZH2rYgB9-6J$. While enumeration is still possible with this method, it would be very time consuming.

5. ATTACKING THE LOCAL DATABASE

In this section we provide an analysis on another attack vector that is usually neglected, but offers huge possibilities for leaking private data. It must be noted that as a prerequisite, access to the local device working directory, as well as to the SD card is a prerequisite, hence this attack is

number	registered	relay	supports_sms	timestamp
+43660	1		0	1431681170252
+4365	1		1	1431681170252

Figure 3: Contact information from the unencrypted contact database of TextSecure

especially interesting in the case of device loss or theft.

5.1 TextSecure

TextSecure uses a user-defined password to encrypt the messages stored in the database. Still, information like contacts and message timestamps are stored unencrypted as shown in Figure 3. This poses a serious problem, as an attacker can still determine the contact information, even in case of intact encryption (see below).

For the message encryption a so called *master secret* is used. This master secret is stored on the device itself, protected by a user defined password. Thus, if the user set no password, the master secret is stored unencrypted and all the messages in the database can be decrypted.

5.2 WhatsApp

The WhatsApp database is stored in plaintext in the directory of the application and only a backup, which is stored on the SD card, is encrypted using AES. Furthermore, the encryption of this backup encounters the same problems as in the TextSecure case: To decrypt this *crypt8 backup database*, only the key from the home directory of the application is required. Following we describe, how to decrypt the backup database of the WhatsApp messenger:

Listing 10 extracts the key and the IV, which are required for the AES decryption, from the key file.

Listing 10: Extraction of the key and the IV

```
k=$(hexdump -e '2/1␣"%02x"' key | cut -b
    253-316)
iv=$(hexdump -e '2/1␣"%02x"' key | cut -b
    221-252)
```

To strip the 67 byte header from the encrypted database, Listing 11 was used.

Listing 11: Stripping of the header

```
dd if=msgstore.db.crypt8 of=msgstore.db.
    crypt8.nohdr ibs=67 skip=1
```

The last step shown in Listing 12 is to decrypt the database with the key, the IV and the encrypted database in the correct format derived from the steps before.

Listing 12: Decryption of the database

```
openssl enc -aes-256-cbc -d -nosalt -
    bufsize 16384 -in msgstore.db.crypt8.
    nohdr -K $k -iv $iv | gunzip > msgstore
    .db
```

Figure 4 shows a snippet from a decrypted database. The sent message, the sender, the recipient, the timestamp of the message and some additional information can be seen there.

p.net	0	1429679910-42	13	0 Ok :)
@g.us	1	1431164220-1	13	0 Hshsbsbs
@g.us	0	1431164441-7	0	0 Gshsjs
@g.us	0	1431164441-8	0	0 Hshsbs
@g.us	0	1431164441-9	0	0 Bsbsbs
@g.us	0	1431164441-10	0	0 Hello world
@g.us	0	1431164441-11	0	0 This is a test
@g.us	1	1431164220-3	13	0 Features overall
@g.us	1	1431164220-4	13	0 Hello world

Figure 4: Messages from the decrypted WhatsApp database

5.3 Line & Telegram

Line and Telegram both do not encrypt the messages stored on the device. For Line the unencrypted database can be found in directory */data/data/jp.naver.line.android/databases*. Also the messages from the private chat are stored unencrypted. Figure 5 shows a short part of the loaded database from the smartphone.

chat_id	from_mid	content	created_time	delivered_time	status
u4485c8292c27	u4485c8292c273b72fdf8eec82d676e87	Hello world	1431435112839	0	1
u4485c8292c27		Gluten	1431435627458	1431435626866	3

Figure 5: Messages from the unencrypted Line database

Something similar holds true for Telegram, here the path for the database is */data/data/org.telegram.messenger/files/cache4.db*.

5.4 WeChat

The messenger WeChat stores all information in an encrypted database. However, only with the knowledge on the system settings of the messenger, the IMEI and the encrypted database, it is possible to retrieve the decryption key and thus to decrypt the database. The system settings contain a unique identifier, the so called *UIN*, which is generated by the messenger. The function to generate the decryption key KEY is defined as $KEY = MD5(IMEI+UIN)[0:7]$ [11]. Thereby, the possible characters range only from a to f and 0 to 9, furthermore only seven characters are used for KEY (see the function definition). Hence it is trivial to brute force the correct combination and get access to the database.

Figure 6 shows some messages from a decrypted database. It contains the sender of each message, the corresponding timestamp, the message itself and further information.

imer	createTime	talker	content	imgPa
	1430403335000	wxid_khqxxx3tzsse21	I've accepted your friend request. Now let's	
	1430455151000	wxid_hjtu8vtxpjut22	To translate a message into system's langua	
	1430455151001	wxid_hjtu8vtxpjut22	你是谁.	
	1430671459000	weixin	Welcome back! Feel free to tell me if you hav	
	1430672090000	wxid_nbzm2uo9uejx22	Blub	

Figure 6: Messages from the decrypted database

The usage of this short key and the circumstance that the key can be generated without any problem thwarts the whole purpose of the encryption since everybody having access to the necessary files, is able to decrypt the database.

6. CONCLUSION

The focus of this work lay on the analysis of popular messengers with respect to providing user privacy. In Section 3 we focused on the privacy features that were provided by the messenger apps themselves, including the privileges required

for installing and operating the respective messengers. Section 4 focussed on attacks against the network metadata, i.e. on the question, whether it was possible to enumerate the user base of a messenger, thus retrieving valuable insight on the messenger user network. This is especially important in case of targeted attacks, e.g. for impersonation of users or for attacks on the authentication mechanisms as devised in [20]. Finally, the topic of device theft brings a new, often neglected, attack vector into the analysis: The protection of user privacy against attackers in the physical possession of the device itself. Here we focused on the internal databases storing the messages.

Regarding future work, we plan on further investigating into the topic of privacy protection in case of stolen or lost devices. Especially the application of more advanced forensic methods against the databases seems to be promising future work. Furthermore, the use of QR-Codes for exchanging contact details could lead to new attack vectors like outlined in [17].

Acknowledgements

This work has been supported by the Austrian Research Promotion Agency under grant 846028 and COMET K1. The financial support by the Austrian Federal Ministry of Science, Research and Economy and the National Foundation for Research, Technology and Development is gratefully acknowledged.

7. REFERENCES

[1] The difficulty of private contact discovery (https://whispersystems.org/blog/contact-discovery). Accessed: 2015-04-26.

[2] Intercept with otr encrypted chat (http://www.spiegel.de/media/media-35552.pdf). Accessed: 2015-04-26.

[3] Perfect forward secret telegram messenger (https://core.telegram.org/api/end-to-end#perfect-forward-secrecy). Accessed: 2015-05-03.

[4] Telegram encryption (https://core.telegram.org/mtproto). Accessed: 2015-04-22.

[5] Textsecure (https://whispersystems.org/blog/the-new-textsecure/). Accessed: 2015-04-22.

[6] Textsecure saying goodbye to encrypted sms/mms (https://whispersystems.org/blog/goodbye-encrypted-sms/). Accessed: 2015-04-22.

[7] Wechat voice privacy policy (http://voice.wechat.com/policy.html), 2013. Accessed: 2015-05-04.

[8] Open whisper systems partners with whatsapp to provide end-to-end encryption (https://whispersystems.org/blog/whatsapp/), 2014. Accessed: 2015-03-07.

[9] J. Bamford. The nsa is building the countrys biggest spy center (watch what you say). *Wired, March*, 15, 2012.

[10] Y. Cheng, L. Ying, S. Jiao, P. Su, and D. Feng. Bind your phone number with caution: automated user profiling through address book matching on smartphone. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, pages 335–340. ACM, 2013.

[11] F. M. Darus. How to decrypt wechat enmicromsg.db database?, 2014. Accessed: 2015-05-03.

[12] N. Davison. Wechat: the chinese social media app that has dissidents worried, 2012. Accessed: 2015-05-04.

[13] B. Eun-ji. Actual users of the messenger line. Accessed: 2015-03-07.

[14] T. Frosch, C. Mainka, C. Bader, F. Bergsma, J. Schwenk, and T. Holz. How secure is textsecure? Cryptology ePrint Archive, Report 2014/904, 2014. `http://eprint.iacr.org/`.

[15] P. Frühwirt, P. Kieseberg, S. Schrittwieser, M. Huber, and E. Weippl. Innodb database forensics: reconstructing data manipulation queries from redo logs. In *Availability, Reliability and Security (ARES), 2012 Seventh International Conference on*, pages 625–633. IEEE, 2012.

[16] G. Greenwald and E. MacAskill. Nsa prism program taps in to user data of apple, google and others. *The Guardian*, 7(6):1–43, 2013.

[17] P. Kieseberg, S. Schrittwieser, M. Leithner, M. Mulazzani, E. Weippl, L. Munroe, and M. Sinha. Malicious pixels using qr codes as attack vector. In *Trustworthy Ubiquitous Computing*, pages 21–38. Atlantis Press, 2012.

[18] A. Mahajan, M. Dahiya, and H. Sanghvi. Forensic analysis of instant messenger applications on android devices. *arXiv preprint arXiv:1304.4915*, 2013.

[19] R. Mueller, S. Schrittwieser, P. Fruehwirt, P. Kieseberg, and E. Weippl. What's new with whatsapp & co.? revisiting the security of smartphone messaging applications. In *Proceedings of the 16th International Conference on Information Integration and Web-based Applications & Services*, pages 142–151. ACM, 2014.

[20] R. Mueller, S. Schrittwieser, P. Fruehwirt, P. Kieseberg, E. Weippl, I. Khalil, and I. Khalil. Security and privacy of smartphone messaging applications. *International Journal of Pervasive Computing and Communications*, 11(2), 2015.

[21] S. Schrittwieser, P. Frühwirt, P. Kieseberg, M. Leithner, M. Mulazzani, M. Huber, and E. R. Weippl. Guess who's texting you? evaluating the security of smartphone messaging applications. In *NDSS*, 2012.

[22] R. Sinha. Line app gets 'hidden chat' feature for encrypted, ephemeral messaging, 2014. Accessed: 2015-03-07.

[23] P. Stirparo and I. Kounelis. The mobileak project: Forensics methodology for mobile application privacy assessment. In *Internet Technology And Secured Transactions, 2012 International Conference for*, pages 297–303. IEEE, 2012.

[24] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith. Sok: Secure messaging. *Security & Privacy Symposium*, 2015.

[25] M. Zweerink. Whatsapp privacy problem explained in detail, 2015. Accessed: 2015-05-17.