Guess Who's Texting You?
Evaluating the Security of Smartphone Messaging Applications

Sebastian Schrittwieser, Peter Frühwirt, Peter Kieseberg, Manuel Leithner,
Martin Mulazzani, Markus Huber, Edgar Weippl
SBA Research gGmbH
Vienna, Austria
(1stletterfirstname)(lastname)@sba-research.org

Abstract

In recent months a new generation of mobile messaging and VoIP applications for smartphones was introduced. These services offer free calls and text messages to other subscribers, providing an Internet-based alternative to the traditional communication methods managed by cellular network carriers such as SMS, MMS and voice calls. While user numbers are estimated in the millions, very little attention has so far been paid to the security measures (or lack thereof) implemented by these providers.

In this paper we analyze nine popular mobile messaging and VoIP applications and evaluate their security models with a focus on authentication mechanisms. We find that a majority of the examined applications use the user's phone number as a unique token to identify accounts, which further encumbers the implementation of security barriers. Finally, experimental results show that major security flaws exist in most of the tested applications, allowing attackers to hijack accounts, spoof sender-IDs or enumerate subscribers.

1 Introduction

In the past few months, several new smartphone messaging and VoIP services with a novel user authentication concept were introduced. These new-generation communication applications aim at replacing traditional text messaging (SMS) and only require the user's phone number for registration. Contrary to well-known instant messaging services, no additional authentication mechanisms other than the phone number are used by these applications. In this paper we focus on the security of applications that are using this novel authentication concept. Due to this limitation, services such as Skype, Facebook Chat and Google Chat were regarded as out of scope. Note that these services have been the subject of an ample amount of past research.

The common advantages of the tools we examined lie in very simple and fast setup routines combined with the possibility to incorporate existing on-device address books. Additionally these services offer communication free of charge and thus pose a low entry barrier to potential customers. However, we find that the very design of most of these messaging systems thwarts their security measures, leading to issues such as the possibility for communication without proper sender authentication.

The main contribution of our paper is an evaluation of the security of mobile messaging applications with the aforementioned properties and the possibilities of abuse in real-world scenarios. Additionally, we draw attention to a number of suitable security mechanisms to prevent the misuse of these systems. The rest of the paper is organized as follows: Section 2 gives an overview of related work. Section 3 outlines the basic functionalities of the examined communications services, while Section 4 introduces our threat assessment for these applications. Section 5 documents our findings and explains how the flaws we identified might pose threats to users. We conclude in Section 6 and give a brief overview of approaches for future research.

2 Related Work

In this paper we document our findings on weak user authentication in messaging applications on smartphones. User authentication is a popular field of research in information security [16, 2], especially applied to distributed systems [13] or for web services [11, 18]. A vast number of protocols has been designed to provide secure user authentication, for example based on Kerberos [15] or public key cryptography and the usage of a PKI [4].

Due to the steadily increasing pervasiveness of smartphones these platforms have sparked the interest of the security community. The security features and properties of

Android [9, 8, 3, 10] as well as iOS [5] have been widely studied. Furthermore, smartphone application security has been evaluated in the past [6, 7]. To the best of our knowledge no evaluation of novel smartphone messaging services analyzed in this paper has been published at the time of writing. Recently, cloud storage services have attracted the interest of security researchers [12] analyzing the implications of faulty authentication in that area. There are numerous applications for Android that promise encrypted, secure communication, such as RedPhone and TextSecure [17].

3 Mobile Messaging Applications

General Characteristics All applications analyzed in this paper have one thing in common: They use the user's phone number as the basis for identification. During the setup process, the software asks the user to enter the phone number of the device. Although Android can grant direct access to the user's phone number to applications, this mechanism is currently not in use. Apple's iOS App Store guidelines on the other hand do not allow applications to access the phone number, making manual input necessary. One major, if unintentional, benefit of this approach is that even devices without a phone module (e.g. a WiFi-only tablet) can be activated using the phone number of another device. It should be noted that these messaging applications use the phone number for user identification only and do not attempt to communicate over the regular mobile phone network. The main problem with this approach is naturally that the system has to verify the user's input, seeing as a malicious user could enter someone else's phone number and therefore hijack or create an account with false credentials.

All the messengers we analyzed implement measures to prevent users from impersonating others by trying to authenticate a number they do not control. Still, several of these approaches display fundamental design flaws. Section 5 analyzes the shortcomings of several messengers.

WhatsApp The most popular tested application (judging by its widespread distribution among various smartphone platforms) is the WhatsApp messenger. It is a cross-platform messaging application for Android, BlackBerry, iOS and Symbian. The vendor has not released any information on its user base, however, based on the Android Market sales, it can be estimated to have at least a few million users[1]. Recently, the vendor reported that in one single day over one billion messages were sent over Whatsapp[2]. In contrast to other comparable messengers, this piece of software does not support calls via VoIP.

[1] https://market.android.com/details?id=com.whatsapp, retrieved on August 23rd, 2011

[2] http://blog.whatsapp.com/index.php/2011/10/one-billion-messages, retrieved on November 2nd, 2011

4 Evaluation

In this section we detail the methodology and the experimental setup of our evaluation.

4.1 Methodology

For our evaluation, we selected nine popular messaging and VoIP applications for both Android and iOS. We estimated the user base of the applications by accumulating data available from the Android Market[3] and Xyologic[4], a company providing download estimations for iOS applications. Table 1 gives an overview of the applications and their features. The great majority of our selected smartphone messaging applications support Voice over Internet Protocol (VoIP) calls and text messages. Furthermore, all tested applications used the user's phone number as the unique user ID for initial authentication, with the Short Message Service (SMS) being the preferred method to verify the user's control over a given phone number. We then identified five possible attack vectors exploiting the insufficient authentication methods employed in these applications. Lastly, we systematically examined the software packages for the presence of these flaws. This section describes the five common attack vectors we identified amongst popular smartphone messaging applications.

Authentication Mechanism and Account Hijacking We analyzed the initial setup mechanisms of the applications during which a phone number is linked to a device. None of the tested applications retrieve the device's phone number automatically but instead ask the user to input it manually during the setup phase. The common method to verify the entered number is sending a SMS message to the specified number containing a verification PIN that the user has to enter in the application's user interface. We analyzed the communication between phone and server during the initial setup and tested if an attacker could hijack accounts by passing another user's phone number as his/her own.

Sender ID Spoofing / Message Manipulation In the second part of our evaluation, we analyzed the communication between the phone and the server during message sending and receiving. The attack scenarios for this part are a malicious user that wants to send a message with a spoofed sender ID. In contrast to the scenario outlined in the previous paragraph, the attacker may do this without hijacking the entire account.

The manipulation of a message during transfer is another possible threat, however, as most tested application use en-

[3] https://market.android.com, retrieved on November 2nd, 2011

[4] http://search.xyologic.com, retrieved on November 2nd, 2011

	VoIP	Text Messages	Number Verification	Uploads Address Book
WhatsApp 2.6.4	no	yes	SMS, active SMS	yes
Viber 2.0.3	yes	yes	SMS and passive phone call	yes
eBuddy XMS 1.15.2	no	yes	SMS	yes
Tango 1.6.9568	yes	no	SMS	yes
Voypi 1.2	yes	yes	SMS	yes
Forfone 1.5.6	yes	yes	SMS	yes
HeyTell 2.3.0	yes	no	no	no
EasyTalk 2.0.1	yes	yes	SMS	yes
Wowtalk 1.0.3	yes	yes	SMS	yes
	Status Messages	Platforms	Estimated User Base	
WhatsApp 2.6.4	yes	Android, iOS, BlackBerry, Symbian	23-63M	
Viber 2.0.3	no	Android, iOS	10-15M	
eBuddy XMS 1.15.2	no	Android, iOS	1-1.5M	
Tango 1.6.9568	no	Android, iOS	10-15M	
Voypi 1.2	no	Android, iOS	0.1-0.15M	
Forfone 1.5.6	no	Android, iOS	0.2-0.25M	
HeyTell 2.3.0	no	Android, iOS	5-9M	
EasyTalk 2.0.1	no	iOS	0.25-0.3M	
Wowtalk 1.0.3	yes	iOS	0.06M	

Table 1. Overview of selected smartphone messaging applications, their features, supported platforms, and estimated user base.

cryption for communication with the server, such an attack would usually not be practical in real life scenarios.

Unrequested SMS/phone calls Most services emit SMS messages or even phone calls throughout the phone number verification process. A malicious user could use another user's number in the setup process to generate annoying messages or phone calls on the victim's phone without revealing his identity.

Another scenario in this class is eavesdropping and replaying a message.

Enumeration Most applications upload the user's address book to the server and compare the entries to a list of registered users (only EasyTalk utilizes a slightly different mechanism and only transmits the number as it is dialed). The server then returns the subset of the user's contacts that are using the service. We analyzed how this mechanism could be used to enumerate users of the service, e.g. by uploading an address book containing a large amount of phone numbers.

The main problem resulting from this functionality is that an attacker can derive useful information about the user's device such as the operating system, if a specific application only runs on one specific system (for instance a certain OS/version combination). This enables the attacker to perform system specific attacks.

Modifying Status Messages Two out of the nine applications allow the user to set a status message that is shared with people that have this user in their address book. In this part of the evaluation, we considered two threats. The first one is the modification of a user's status message by an attacker. We analyzed the protocol for setting the status message and explore possible vulnerabilities that could result in unauthorized modification of status messages.

The second threat is a privacy-related design error. Not only is it possible to determine whether the owner of a given phone number has installed the messenger application (as outlined above), but also the status message of a user is visible to people that have stored this user in their address book. Since no user confirmation is required to store a number in the address book, an attacker can very easily get access to the status messages of all subscribers to services vulnerable to this attack. In practice, this approach would likely be combined with some sort of enumeration attack.

4.2 Experimental Setup

For our security evaluation we used a Samsung Nexus S running Android 2.3.3 and an Apple iPhone 4 running iOS 4.3.3. Applications that are available for both platforms were tested on both the Nexus S and the iPhone. To be able to read encrypted HTTPS traffic from and to the tested applications, we set up a SSL proxy that acted as a man-in-the-middle and intercepted requests to HTTPS servers. We further used SSLsniff [14] by Moxie Marlinspike to read SSL-

protected traffic that is not sent over HTTPS (e.g. XMPP).

Figure 1 explains our approach for the experimental setup. The SSL proxy was used to analyze HTTPS connections and allowed us to read as well as modify HTTPS traffic on the fly. Other protocols were observed with SSLsniff.

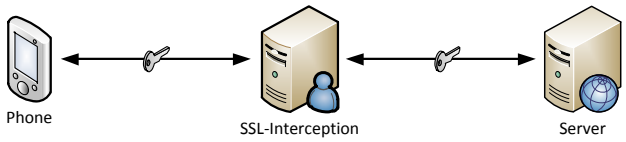

Figure 1. Experimental setup for intercepting SSL.

5 Results

In this section we present the results of our security evaluation of the application discussed in Section 3 with respect to the different attacks outlined in section 4.

5.1 Overview

Table 2 gives a compact overview of the vulnerabilities found in the tested applications. It is notable that almost all applications were vulnerable to SMS flooding and enumeration attacks, but only very few to sender spoofing or message manipulation.

5.2 Authentication Mechanism and Account Hijacking

In this section we describe successful attacks against the authentication mechanisms of the tested applications. The general idea is that an attacker tries to hijack accounts to be able to spoof the sender ID and receive messages targeted to a victim. In essence, the attacker aims at linking his mobile device to the phone number of the victim.

WhatsApp To prevent malicious users to impersonate somebody else using the victim's number, a verification SMS containing a 4-digit PIN is sent to the phone. The user then has to copy that code into the WhatsApp application's GUI. This process binds a WhatsApp user account (represented by the phone number) to a physical device.

Figure 2 shows the authentication process of WhatsApp. We discovered that the verification process of WhatsApp is fatally broken. The PIN for the verification SMS message is generated on the phone and then sent to the server via a HTTPS connection. The server then initiates the SMS message via a SMS proxy to the phone, where the app then

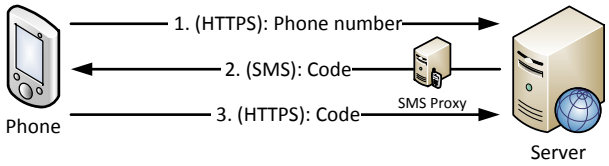

Figure 2. Authentication process of WhatsApp

checks if the PIN entered by the user matches the previously generated PIN. An attacker could exploit this mechanism to hijack any WhatsApp account. This can be done by typing the victim's phone number during the verification phase and then intercepting the communication between the phone and the server to eavesdrop the PIN. This communication is SSL-protected; however, the attacker has to intercept only the connection between his own phone and the WhatsApp server. To exploit this vulnerability, it is possible set up a SSL proxy and install the proxy's certificates as described in Section 4 on the phone in order to get access to the encrypted communication transparent to the application.

Once the attacker has entered the PIN into his phone, the victim's WhatsApp account is linked to the attacker's phone. This enables the attacker to send and retrieve messages from the victim's account. This process also unlinks the victim's device, causing it to not receive messages from WhatsApp anymore.

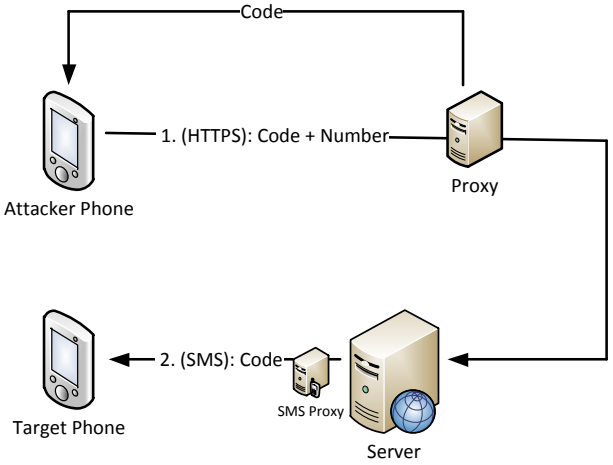

Figure 3. MitM-Attack against WhatsApp authentication

Figure 3 shows a possible attack on the authentication process of WhatsApp. A man-in-the-middle attack on the communication between the phone and the client makes it possible to eavesdrop the secret SMS verification code be-

	Account Hijacking	Spoofing / Manipulation	Unrequested SMS	Enumeration	Other Vulnerabilities
WhatsApp	yes	no	yes	yes	yes
Viber	no	no	yes	yes	
eBuddy XMS	no	no	yes	yes	
Tango	yes	no	yes	yes	
Voypi	yes	yes	yes	yes	yes
Forfone	no	yes	yes	yes	
HeyTell	yes	no	no	limited	
EasyTalk	yes	no	yes	yes	
WowTalk	yes	no	yes	yes	yes

Table 2. Overview on attacks.

fore it was even delivered to the spoofed phone number.

Tango and Voypi The applications Tango and Voypi share a very similar approach for device registration. Like WhatsApp, both applications ask the user to enter the device's phone number. If the number is not registered for the service yet, no verification is done. Only if the number is already known to the system, a verification process via SMS (similar to WhatsApp) is performed.

While this registration schema is not vulnerable to account hijacking, an attacker can impersonate users that are not yet registered for that service. As long as a number is not registered for Tango or Voypi, an attacker can use it without SMS verification.

HeyTell HeyTell does not have any kind of verification. During the setup process the user has to select his or her own cellphone number from the address book (or create a new entry if it does not exist). The device is then linked to the chosen number without verification.

WowTalk WowTalk's registration mechanism is based on SMS-verification. The user has to enter his phone number into the application which transmits it to the server. The server generates a random verification code and sends it back to the phone via SMS. The problem, however, is that the server also sends the verification code via HTTPS to the phone so that it can compare the user's input to the correct code. We used the SSL proxy to intercept the server's reply and so retrieve the verification code. An attacker can use this technique to hijack any WowTalk account. Figure 4 explains our attack against WowTalk's client authentication.

EasyTalk EasyTalk uses SMS for phone number verification. After a device's registration request, the server generates a verification code that is sent to the device via SMS. After receiving the SMS the user has to enter the code into the application that forwards it to the server for verification. The server then replies to the device with either "OK" if the device sent the correct code or "ERROR" if the user entered

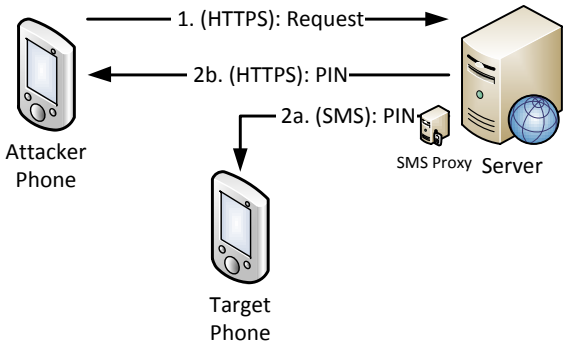

Figure 4. MitM-attack against WowTalk application

an incorrect code into the application. We were able to successfully authenticate a client by modifying this message from "ERROR" to "OK". The server does not detect this message manipulation and keeps the device authenticated.

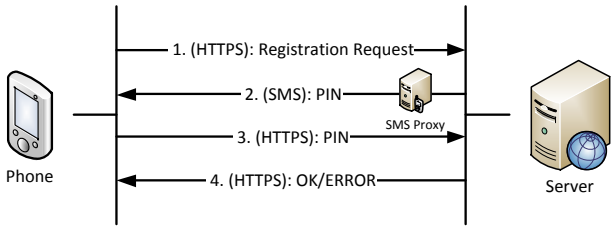

Figure 5. Device authentication in EasyTalk.

Figure 5 shows the authentication mechanism of EasyTalk.

Viber Compared to the other introduced applications, Viber's authentication mechanism is well designed and properly implemented. The application asks the user for the phone number and sends an authentication request to the server. The server generates a verification code and sends it via SMS message to the user's phone. Alternatively, the

user can request a phone call from Viber. In that case, a speech synthesizer voice speaks the code on the phone call. The user has to the enter the received code in the Viber application that forwards it to the server, which in turn checks the input. At no time does the server trust the client (i.e. the application on the user's phone) and no sensitive authentication data is transmitted between phone and server. Figure 6 explains the authentication mechanism of Viber.

Forfone and eBuddy XMS The authentication mechanisms of Forfone and eBuddy XMS are similar to Viber's and thus not susceptible to attacks that are based on intercepting the communication between the device and the server.

Conclusion We do not propose our own authentication schema, as some of the tested applications already have secure protocols. While a secure implementation seems trivial, our evaluation showed that the majority of the tested applications are susceptible to even basic attacks.

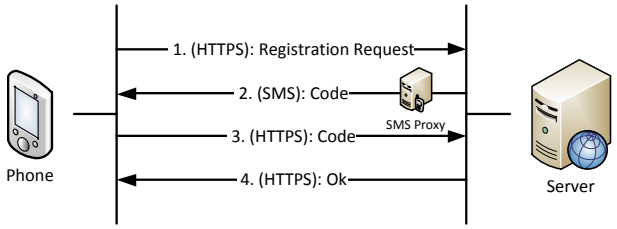

Figure 6. Authentication in Viber.

5.3 Sender ID Spoofing

This section introduces the results of our evaluation of messaging protocols in the tested applications. We analyzed the protocols and attempted to send messages with spoofed sender IDs without hijacking the entire account. Most of the tested applications use the Extensible Messaging and Presence Protocol (XMPP) [19] for messaging and can therefore rely on the security features present in the XMPP server that prevent sender ID spoofing. However, Voypi and Forefone have their own implementations for messaging that are based on HTTP(S) requests.

Voypi The unencrypted HTTP request that is used by Voypi to generate messages has four GET parameters: both the sender's and the receiver's phone number, the message, and a time stamp. There is no authentication required to send a message, therefore, an attacker can spoof the sender ID.

Forfone In Forfone's messaging protocol an additional identifier of the sender is required for sending messages. In Android the IMSI of the phone and in iOS the UDID (Unique Device Identifier) are used for sender authentication. While this additional parameter raises the difficulty of sender ID spoofing, it cannot be considered as a secure authentication mechanism as these two identifiers can be accessed by any third party application on the phone.

Table 3 summarizes the layout of the messaging protocols used by Voypi and Forfone.

	Voypi	Forfone
HTTP-Method	GET	POST
Parameters	sender phone number	sender phone number
	receiver phone number	receiver phone number
	receiver country code	
	message	message
	timestamp	sender UDID

Table 3. Messaging protocols of Voypi and Forfone.

5.4 Unrequested SMS

When taking a look at the authentication mechanisms, several applications use an SMS sent to the (assumed) requesting handset in order to verify the validity of the request and to thwart account hijacking (see Section 5.2). However, several implementations of this mechanism can be misused in order to send these verification-messages to the arbitrary users. In one application (WhatsApp) we were even able to modify the messages sent, thus being able to communicate via SMS worldwide free of charge.

General Approach The general approach consists of spamming an arbitrary user with text messages containing validation requests from a messaging service. Since we are able to do this in an automated fashion, this can be frustrating to a victim constantly receiving authorization requests. Unfortunately for the attacker (and fortunately for potential victims) all examined applications had some kind of timeout that thwarted real mass spamming. Still, an attacker is able to send messages at a regular interval. The adversary could also target certain numbers used for emergencies such as those used by system administrators in high-availability data centers, potentially causing the victim to switch off her device. The idea behind the attack can be seen in Figure 7

This approach works for the following messengers: WhatsApp, Viber, Tango, eBuddy XMS, WowTalk, Voypi, Forfone and easyTalk. However, no application other than WhatsApp allow for the injection of content into the verification SMS, thus making it rather useless for actual spamming (as used for marketing purposes).

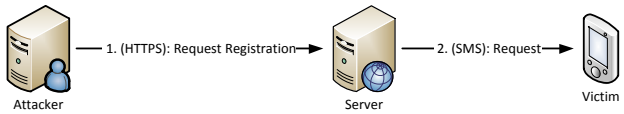

Figure 7. General approach.

Viber The messaging application Viber allows for an even more annoying attack. In case the SMS message for authentication was not answered by the target, the attacker can choose to set up an authentication request using a phone call (Figure 8 shows the relevant parts of the authentication mechanism in Viber).

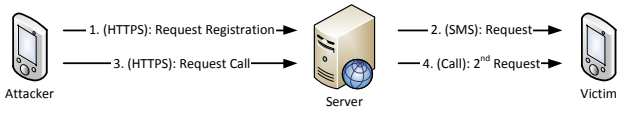

Figure 8. Sending SMS and requesting phone calls with Viber.

WhatsApp Finally, WhatsApp offers an intriguing feature (or rather design error) that can be (mis-)used for sending free text messages worldwide. When looking at the authentication mechanism, we discussed that the authentication code is chosen by the handset that is requesting authentication and the WhatsApp-server simply echoes it back to the handset by a SMS. This authentication code consists of the string "WhatsApp code" followed by a PIN generated by the phone. In our analysis, we were able to intercept and modify the transmission of the PIN from the phone to the server. The modified SMS was even delivered when the PIN was replaced by any alphanumerical string, allowing an attacker to send SMS messages with nearly arbitrary content to the target handset. These techniques can either be used for free communication, or for sending spam. In fact, the verification messages can be sent by requesting the URL

```
https://s.whatsapp.net/client/iphone/
      smsproxy.php?to=\<receiver\>
      &auth=\<text message\>.
```

Figure 9 details the relevant parts of the authentication protocol that are abused in this attack.

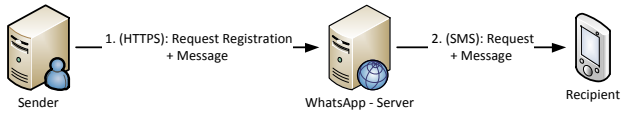

Figure 9. Free SMS with WhatsApp

5.5 Enumeration

Another security-relevant aspect of this type of messaging applications is their ability to automatically import the user's contacts. All tested applications except HeyTell allow the user to upload the entire address book to the system's server and compare the contained phone numbers to already registered phone numbers stored on the server. The server returns a subset of the user's contact list containing only phone numbers that are registered.

A possible threat resulting from a user account enumeration is the identification of active phone numbers. Furthermore, an attacker could try to identify the operating system of the user, based on the applications installed on the phone and their availability for a certain operating system. This enables an attacker to target a system with OS-specific exploits.

We tested the feasibility of such an enumeration attack with WhatsApp. To this end, we selected the US area code 619, which covers the southern half of the city of San Diego, CA and enumerated the entire number range from 000-0000 to 999-9999. A similar approach for Facebook is described by Balduzzi et al. [1]. In their paper, the authors tested the validity of mail addresses by uploading them to the friend-finder feature of Facebook. Based on the return value of Facebook, they were able to determine the status of a mail address.

In our evaluation, we split the entire number range of the San Diego area code 619 into chunks of 5000 phone numbers each and simulated a standard address book upload as performed by WhatsApp during device registration. While we noticed some slowdowns in server response time during our evaluation, the WhatsApp server did not prevent us from uploading ten million phone numbers and returned 21095 valid phone numbers that are using the WhatsApp application as well as their status messages. The entire process finished in less than 2.5 hours.

Figure 10 shows the distribution of phone numbers that are linked to a WhatsApp account over the entire number range. As the figure indicates, active phone numbers of the area code 619 start at 200000. We believe that the mobile number range starts above this value, but have not independently confirmed that.

A simple countermeasure would be the introduction of a rate limit. Clearly, no regular user would upload ten million phone numbers and such an attempt could be easily detected and blocked by the server.

HeyTell HeyTell does not support upload of a whole address book for enumeration, but enumeration can be done number by number by requesting to send a voice message for every single number in the address book. This, however, is restricted by a privacy setting that allows users to

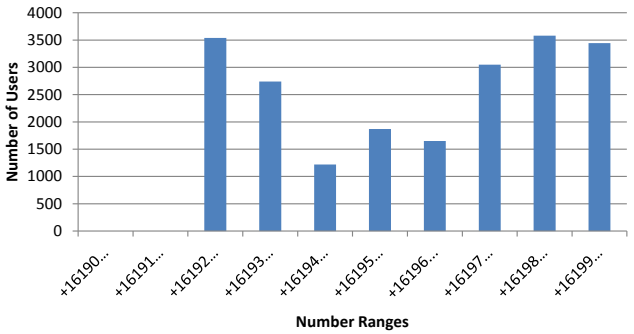

Figure 10. Distribution of phone numbers in area code 619 that are registered with WhatsApp.

limit their visibility.

5.6 Other Vulnerabilities

WhatsApp An additional feature of WhatsApp is the possibility to set a status message, similar to instant messaging clients like Skype, that can be read by the user's contacts. Changing this status message does not require any authentication. In fact, everyone can change anyone else's status message by sending an HTTPS request to

```
https://s.whatsapp.net/client/iphone/u.php?
      cc=<country code>&me=<phone number>
      &s=<status message>.
```

WowTalk Like WhatsApp, WowTalk offers the feature of setting a status message. A user can change the status message for an arbitrary user by issuing a POST-request to `https://sip.wowtalk.org/wowtalk_srv.php` containing

```
action=user_status_update&username=<user id>
     &status=<New Status>
```

Voypi We were able to identify two other vulnerabilities in Voypi. It is possible to request Voypi users in the address book of other users. To this end, a simple HTTP request with the phone number of the victim is sent to the server:

```
http://msg.voypi.com/myphone_v1/getusers.php?
     phone=<phone number>&version=1.2
```

No further authentication is required to perform this query. The server responds with the subset of Voypi users from the victim's address book containing names and phone numbers.

The second vulnerability allows an attacker to request messages of other users without authentication. The Voypi client has to be running in order to be able to receive messages. When active, Voypi pulls new messages in an interval of seven seconds from the server:

```
http://msg.voypi.com/myphone_v1/getmsg.php?
     dbname=<phone number>8&version=1.2
```

As long as the client has not pulled messages, an attacker can do so and thus steal another user's messages. The term "stealing" fits this scenario, because the victim is not able to retrieve the messages from the server once the attacker has done so.

6 Conclusion

In this paper, we assessed nine mobile messaging and VoIP applications for smartphones. Our evaluation showed that many applications have broken authentication mechanisms and thus are vulnerable to account hijacking attacks. Most applications also suffer from other vulnerabilities such as account enumeration. We practically demonstrated an attacker's capability to enumerate any number of active WhatsApp accounts with a given area code (US area code 619 in our example, which corresponds to Southern San Diego, CA). All identified flaws stem from well-known software design and implementation errors. Although these vulnerabilities may not endanger human lives, they might have a severe impact on the privacy of millions of users.

Future work might include security assessments of upcoming solutions slated for mass adoption such as Apple's iMessage. Furthermore, research towards an authentication scheme suitable as a best practice template for newly developed applications would be a welcome addition.

7 Acknowledgements

This work has been supported by the Austrian Research Promotion Agency under grant 824709 (Kiras) and the Austrian COMET Program (FFG).

References

[1] M. Balduzzi, C. Platzer, T. Holz, E. Kirda, D. Balzarotti, and C. Kruegel. Abusing social networks for automated user profiling. In *Recent Advances in Intrusion Detection: 13th International Symposium, RAID 2010, Ottawa, Ontario, Canada, September 15-17, 2010, Proceedings*, volume 6307, page 422. Springer-Verlag New York Inc, 2010.

[2] M. Bishop. *Computer Security: Art and Science*. Addison-Wesley, 2002.

[3] L. Davi, A. Dmitrienko, A. Sadeghi, and M. Winandy. Privilege escalation attacks on android. *Information Security*, pages 346–360, 2011.

[4] W. Diffie and M. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976.

[5] M. Egele, C. Kruegel, E. Kirda, and G. Vigna. Pios: Detecting privacy leaks in ios applications. In *Network and Distributed System Security Symposium (NDSS)*, 2011.

[6] W. Enck, P. Gilbert, B. Chun, L. Cox, J. Jung, P. McDaniel, and A. Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, pages 1–6. USENIX Association, 2010.

[7] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri. A study of android application security. In *Proc. of the 20th USENIX Security Symposium*, 2011.

[8] W. Enck, M. Ongtang, and P. McDaniel. On lightweight mobile phone application certification. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 235–245. ACM, 2009.

[9] W. Enck, M. Ongtang, and P. McDaniel. Understanding Android Security. *Security & Privacy, IEEE*, 7(1):50–57, 2009.

[10] A. Felt, H. Wang, A. Moshchuk, S. Hanna, E. Chin, K. Greenwood, D. Wagner, D. Song, M. Finifter, J. Weinberger, et al. Permission re-delegation: Attacks and defenses. In *20th Usenix Security Symposium, San Fansisco, CA*, 2011.

[11] K. Fu, E. Sit, K. Smith, and N. Feamster. Dos and Don'ts of Client Authentication on the Web. In *Proceedings of the 10th conference on USENIX Security Symposium-Volume 10*, pages 19–19. USENIX Association, 2001.

[12] Intrepidus Group. Intrepidus group, 2011. [Online; retrieved Aug 21st, 2011], `http://intrepidusgroup.com/insight/2011/08/dropbox-for-android-vulnerability-breakdown/`.

[13] B. Lampson, M. Abadi, M. Burrows, and E. Wobber. Authentication in distributed systems: Theory and practice. *ACM Transactions on Computer Systems (TOCS)*, 10(4):265–310, 1992.

[14] M. Marlinspike. Website of sslsniff tool, 2011. [Online; retrieved Jun 21st, 2011], Online at `http://www.thoughtcrime.org/software/sslsniff`.

[15] B. Neuman and T. Ts'o. Kerberos: An authentication service for computer networks. *Communications Magazine, IEEE*, 32(9):33–38, 1994.

[16] W. Stallings. *Cryptography and network security: principles and practice*. Prentice Hall Press, 2010.

[17] Whisper Systems. Whisper systems, 2011. [Online; retrieved Aug 21st, 2011], `http://www.whispersys.com/`.

[18] A. Whitten and J. Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *Proceedings of the 8th USENIX Security Symposium*, pages 169–184, 1999.

[19] XMPP Foundation. XMPP Standard, 2011. [Online; retrieved Jun 21st, 2011], `http://xmpp.org/1`.