Dearth the Security of Smartphone Messaging Application: WhatsApp

Kritarth Y. Jhala[1] Darsh Patel[2]
[1,2]Department of Digital Forensics Analyst
[1,2]eSF Labs Ltd. Hyderabad, India

Abstract— The domination of social networking applications such as WhatsApp, Facebook, Viber continues to grow exponentially with WhatsApp being the undisputed leader amongst a vast series of social networking and chat messengers with more than 600 million users worldwide and being the number one paid application in more than 131 countries. Arguably privacy is a key draw for social media such as facebook, viber, whatsapp users when it comes to any mobile chat option and as a leader on the market WhatsApp has found itself in some hot water after researchers called out a potential security vulnerability. Generally WhatsApp is a very popular messenger for smart devices only. This paper demonstrates how to capture the password of targate device WhatsApp account using the different gateway level proxies like SSL/TLS proxy, MITM proxy. Once attacker received this password attacker can use it to communicate to the WhatsApp main servers directly or via a local framework.

Key words: Whatsapp Sniffer, MITM Attack in WhatsApp, Whatsapp Hacking, MITM attack in mobile applications, Remote Whatsapp Password Cracking

I. INTRODUCTION

The rapid evolution in the Smartphone technologies the development industry has seen a sudden surge in the number of applications available for a Smartphone user. While there are many applications that cater to every one's needs social networking applications occupy a lion's share in the number of users who frequently use them on a daily basis.

General Characteristics All applications analysed in this paper have one thing in common: Every application is used in the user's phone number as the basis for authentication & identification. During the installation process the application asks the user credentials like user phone number. Though 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 create an account with false credentials.

All the messengers we analysed 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.

WhatsApp the most popular tested application deciding by its extensive 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 different app market sales it can be estimated to have at least a few million users. Recently, the vendor reported that in one single day over one billion messages were sent over Whatsapp. In contrast to other comparable messengers this piece of software does not support calls via VoIP.

II. 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 especially applied to distributed systems or for web services .A vast number of protocols has been designed to provide secure user authentication, for example based on Kerberos or public key cryptography and the usage of a PKI

Due to the steadily increasing Android and iOS have been extensively studied. Additionally, application security of smartphone has been calculated in the past. 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 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.

III. EXPERIMENTAL SETUP

For our security evaluation we used a Samsung S2 running Android 4.3.3 and an Apple iPhone 4s running iOS 8.3.3. Applications that are available for both platforms were tested on both the Samsung 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 by Moxie Marlinspike to read SSL-protected traffic that is not sent over HTTPS 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.

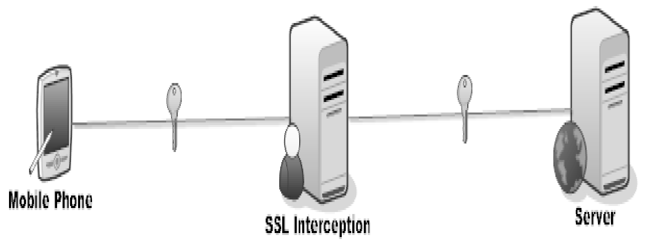


Fig. 1: Experimental setup for intercepting SSL.

A. *Software Tools:*

1) Android 4.0 Operating System
2) WhatsApp Application (Version 4.0.0)
3) Parrot Operating System

B. *Hardware:*

1) Workstation (Intel i5, 8GB DDR3 RAM, 1 TB Seagate HDD, Windows 8 OS with Virtual Machine)
2) USB Data Cable
3) Samsung s2 i9100

IV. METHODOLOGY

WhatsApp For our experiment we selected some popular messaging and VoIP applications for both Android and iOS. The great majority of our selected smart-phone 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.

A. *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.

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 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 installs 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.

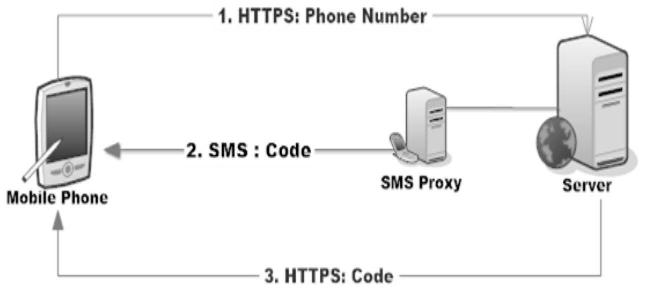


Fig. 2: Authentication process of WhatsApp

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.

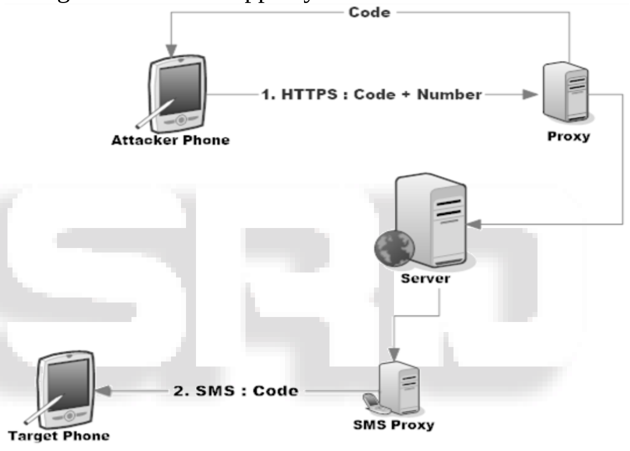


Fig. 3: MitM-Attack against WhatsApp authentication

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 before it was even delivered to the spoofed phone number.

B. *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 encryption 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 an-other 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 re-playing a message.

Enumeration Most applications upload the user's address book to the server and compare the entries to a list of registered users. 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.

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. 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.

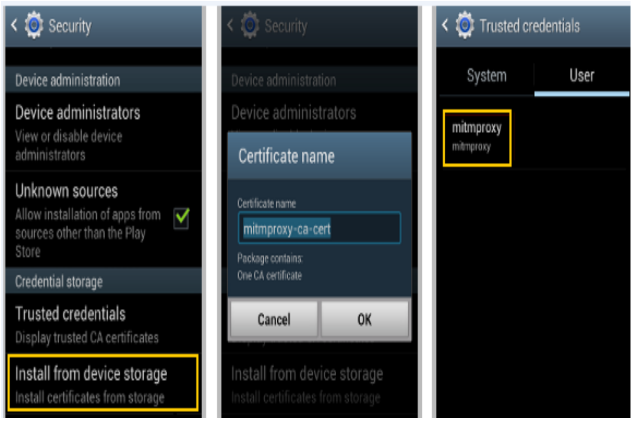
The second threat is a privacy-related design error. Not only is it possible to determine whether the owner of a given phone number has installed the messenger application but also the status message of a user is visible to people that have stored this user in their address book. Since no user confirmation 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.

V. Results

This section describes a man in the middle attack to intercepting the conversation between target device and WhatsApp servers.

A. *Set Up A MITM Proxy to the Attacker Device:*

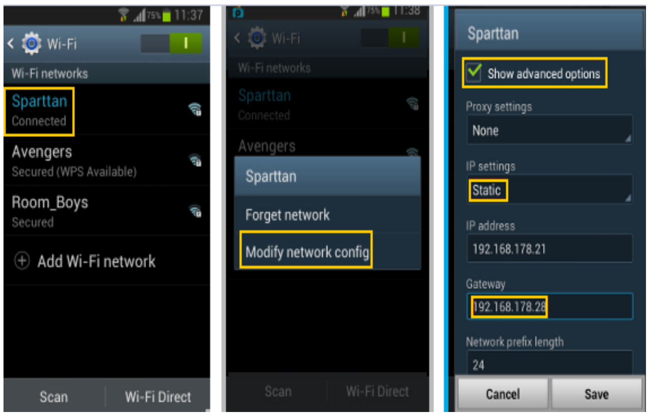
The first step to perform a setup of mitmproxy. For setting up a mitmproxy we have to setup an mitmproxy CA certificate to the attacker device. Figure shows the setup of CA certificate in attacker's handled device


Security
Device administration
Device administrators
View or disable device administrators
Unknown sources
Allow installation of apps from sources other than the Play Store
Credential storage
Trusted credentials
Display trusted CA certificates
Install from device storage
Install certificates from storage
Certificate name
Cancel
OK
Trusted credentials
System
User
mitmproxy
mitmproxy

Certificate name: mitmproxy-ca-cert
Package contains: One CA certificate

Fig. 4: Installation of mitmproxy CA certificate

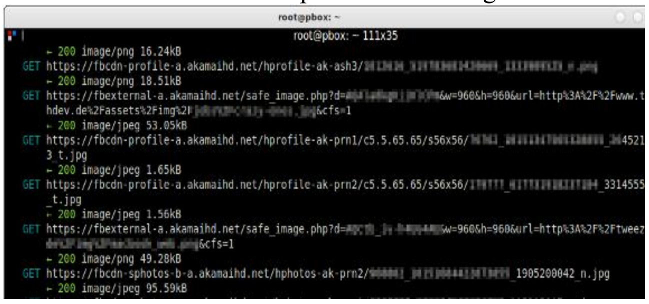
Attackers have to redirect all the traffic from target device to to attackers' linux machine running mitmproxy by changing default gateway of the targated device. Figure 5 shows changing standard gateway of attacker's device.



Proxy settings: None
IP settings: Static
IP address: 192.168.178.21
Gateway: 192.168.178.28
Network prefix length: 24

Fig. 5: Changing of standard gateway of attacker's device

If everything is done correctly an attacker can see all the traffic that attacker's phone sends as fig 6

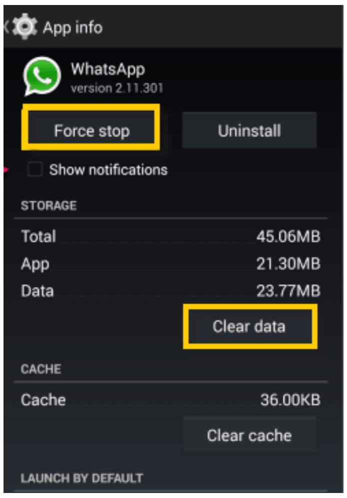


```
root@pbox: ~
root@pbox: ~ 111x35
    ← 200 image/png 16.24kB
GET https://fbcdn-profile-a.akamaihd.net/hprofile-ak-ash3/[redacted]
    ← 200 image/png 18.51kB
GET https://fbexternal-a.akamaihd.net/safe_image.php?d=[redacted]&w=960&h=960&url=http%3A%2F%2Fwww.t
    hdev.de%2Fassets%2Fimg%2F[redacted]&cfs=1
    ← 200 image/jpeg 53.05kB
GET https://fbcdn-profile-a.akamaihd.net/hprofile-ak-prn1/c5.5.65.65/s56x56/[redacted]4521
    3_t.jpg
    ← 200 image/jpeg 1.65kB
GET https://fbcdn-profile-a.akamaihd.net/hprofile-ak-prn2/c5.5.65.65/s56x56/[redacted]_3314555
    _t.jpg
    ← 200 image/jpeg 1.56kB
GET https://fbexternal-a.akamaihd.net/safe_image.php?d=[redacted]&w=960&h=960&url=http%3A%2F%2Ftweez
    [redacted]&cfs=1
    ← 200 image/png 49.28kB
GET https://fbcdn-sphotos-b-a.akamaihd.net/hphotos-ak-prn2/[redacted]_1905200042_n.jpg
    ← 200 image/jpeg 95.59kB
```

Fig. 6: Whatsapp Traffic after setup

B. *Clear User Data & Stop Whatsapp:*

WhatsApp only adjudicate a new password with the server when application first communicates with server. If attacker device already have WhatsApp application up and running on attacker's Android or IOS devices attacker need to wipe the data. Therefore WhatsApp can negotiate with a new password which attacker can easily sniff using mitmproxy. here fig. 7 describe how to wipe all user data of the application.



Total	45.06MB
App	21.30MB
Data	23.77MB
Cache	36.00KB

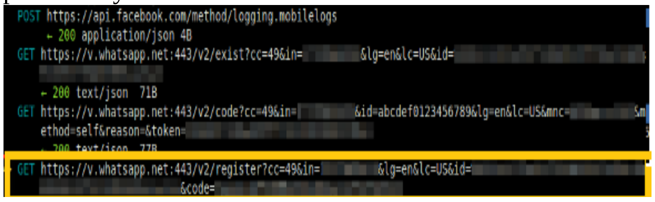
Fig. 7: Clear whatsapp Data

C. *Restore Data & Sniff the Password.*

After wiping all WhatsApp data now it's like fresh application. Now when attacker opens a WhatsApp it has to reconnect to the WhatsApp account and replace a new login password. By given that attacker is now has a sniffer in place and it can also pull all of target devices communication attacker can simply read the password. Make sure that mitmproxy is running.

D. Results of MITM Proxies:

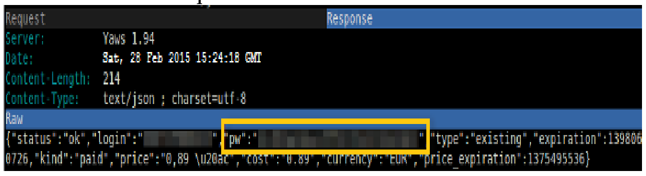
Figure shows requests to the domain v.whatsapp.net particularly.



```
POST https://api.facebook.com/method/logging.mobilelogs
   ← 200 application/json 4B
GET https://v.whatsapp.net:443/v2/exist?cc=49&in=[redacted]&lg=en&lc=US&id=[redacted]
    [redacted]
  ← 200 text/json  71B
GET https://v.whatsapp.net:443/v2/code?cc=49&in=[redacted]&id=abcdef0123456789&lg=en&lc=US&mnc=[redacted]&m
    ethod=self&reason=&token=[redacted]
  ← 200 text/json  77B
GET https://v.whatsapp.net:443/v2/register?cc=49&in=[redacted]&lg=en&lc=US&id=[redacted]
    [redacted]&code=[redacted]
```

Fig. 8: MITM Proxies

Now attacker can find password of target account in the last three requests.



```
Request                                   Response
Server:          Yaws 1.94
Date:            Sat, 28 Feb 2015 15:24:18 GMT
Content-Length:  214
Content-Type:    text/json ; charset=utf-8
Raw
{"status":"ok","login":"[redacted]","pw":"[redacted]","type":"existing","expiration":139806
0726,"kind":"paid","price":"0,89 \u20ac","cost":"0.89","currency":"EUR","price_expiration":1375495536}
```

Fig. 9: Password Sniffing

Copy that password and save the conversation with the WhatsApp server by using mitmproxy's save function

VI. CONCLUSION

In this paper, we assessed mobile messaging and VoIP applications for smartphones like whatsapp. Our research showed that popular application in the area of messaging & calling have broken authentication mechanisms and thus are vulnerable to account hijacking attacks. This application also suffers 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. 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.

VII. ACKNOWLEDGMENT

This work was supported by eSF Labs Ltd, Hyderabad, India, that provided the technical conditions and the machines used for the development and testing of the solution.

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, On-tario, 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] Sniff the WhatsApp password: http://blog.philippheckel.com/2013/07/05/how-to-sniff-the-whatsapp-password-from-your-android-phone-or-iphone/