Forensics study of IMO call and chat app

Article *in* Digital Investigation · April 2018

DOI: 10.1016/j.diin.2018.04.006

CITATIONS
0

READS
20,417

5 authors, including:

K. Sudozai
National University of Sciences and Technology
4 PUBLICATIONS **1** CITATION
SEE PROFILE

Shahzad Saleem
National University of Sciences and Technology
23 PUBLICATIONS **86** CITATIONS
SEE PROFILE

William J Buchanan
Edinburgh Napier University
461 PUBLICATIONS **1,042** CITATIONS
SEE PROFILE

Haleemah Zia
National University of Sciences and Technology
5 PUBLICATIONS **31** CITATIONS
SEE PROFILE

Digital Investigation 25 (2018) 5–23

Contents lists available at ScienceDirect

Digital Investigation

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

Forensics study of IMO call and chat app

M.A.K. Sudozai [b], Shahzad Saleem [b], William J. Buchanan [a, *], Nisar Habib [b], Haleemah Zia [b]

[a] The Cyber Academy, Edinburgh Napier University, Edinburgh, UK
[b] School of Electrical Engineering and Computer Science (SEECS), National University of Sciences and Technology (NUST), Islamabad, 44000, Pakistan

ARTICLE INFO

Article history:
Received 4 January 2018
Received in revised form
17 April 2018
Accepted 18 April 2018
Available online 25 April 2018

Keywords:
IMO
Encryption
Android
iOS
Network forensic
Device forensic

ABSTRACT

Smart phones often leave behind a wealth of information that can be used as an evidence during an investigation. There are thus many smart phone applications that employ encryption to store and/or transmit data, and this can add a layer of complexity for an investigator. IMO is a popular application which employs encryption for both call and chat activities. This paper explores important artifacts from both the device and from the network traffic. This was generated for both Android and iOS platforms. The novel aspect of the work is the extensive analysis of encrypted network traffic generated by IMO. Along with this the paper defines a new method of using a firewall to explore the obscured options of connectivity, and in a way which is independent of the protocol used by the IMO client and server. Our results outline that we can correctly detect IMO traffic flows and classify different events of its chat and call related activities. We have also compared IMO network traffic of Android and iOS platforms to report the subtle differences. The results are valid for IMO 9.8.00 on Android and 7.0.55 on iOS.

Introduction

IMO (http://imo.im) is a free messaging, voice and video call application (app) and which was launched in 2007 by Ralph Harik, Georges Harik and Praveen Krishnamurthy (Crunchbase, May 23, 2013 [accessed 16-March-2017]; Eldon, 2007 [accessed 16-March-2017]). According to a survey conducted in 2016 by App Annie, IMO is one of the top communication apps being used worldwide (AppAnnie, 2016 [accessed 16-March-2017]). Apart from its user-friendly design and reliable connectivity, one major factor that contributes to this popularity is its provision of service in countries where competitor apps (WhatsApp or Viber) are blocked by government agencies (Quora, May, 2016 [accessed 17-March-2017]; ProVpnAccounts, May, 2016 [accessed 17-March-2017]).

Our major contribution in this paper is the exploration and analysis of IMO artifacts for:

- The artifacts generated through the usage on of mobile device.
- The artifacts generated through usage over the network.
- An investigation of both Android and iOS devices.

The results of mobile device forensics indicate that IMO stores data in plain text, so that anyone with the control of the smart phone can have access to the underlying data. We have conducted a detailed study of IMO file structure on both Android and iOS platforms and define the grey areas which can be exploited during the forensics study of IMO.

From the network perspective, it is important to highlight that the communication protocol of IMO, as well as its security architecture, are not known in public literature. We have performed an extensive review of the traffic analysis of IMO and have introduced the idea of incorporating a firewall approach to the investigation. The firewall helps understand the patterns of connectivity and then can regulate the traffic based on a progressive study. We thus forced the IMO client to connect to its servers in a controlled environment and this arrangement revealed the obscured design of IMO client connectivity to its servers. After this, we experimented with the network activities of IMO on both the platforms in order to study different traffic characteristics.

In Section Related work, we have summarized the previous work done in forensics analysis of social media apps. Section Device forensics of IMO covers the device forensics part of IMO, along with our analysis methodology, detailed experimental setup for accessing the Android and iOS storage/memory and results of our study have been separately mentioned for both these platforms. Section Network forensics of IMO defines the network forensics elements of IMO, and discuss the traffic analysis setup and the results for both Android and iOS platforms. Section Crime scene reconstruction

* Corresponding author.
E-mail address: w.buchanan@napier.ac.uk (W.J. Buchanan).

https://doi.org/10.1016/j.diin.2018.04.006
1742-2876/

demonstrates the application of our results to reconstruct a crime scene involving IMO. The paper is finally concluded in Section Conclusion.

Related work

Over the past few years, users concerns over their privacy has been increasing, alongside the number of social media apps providing privacy to their users (Taylor et al., 2014). Besides their positive use, the secure services offered by these apps are also extensively exploited in variety of criminal cases. Digital forensics thus has become an most important component of any crime investigation (Seigfried-Spellar and Leshney, 2015; Huber et al., 2011; Brunty et al., 2014). For any mobile social media app, forensics analysis has two dimensions:

- Device forensics. This includes analyzing memory and storage elements (Norouzizadeh Dezfouli et al., 2016).
- Network forensics. This involves the study of network traffic for different activities of users and services (Lillard, 2010).

The most comprehensive study of network and device forensics of Android social-messaging applications has been carried out by (Walnycky et al., 2015). This includes the forensics analysis of 20 of the most popular social media apps for Android. The authors highlight the features of target apps and which leave artifacts of evidentiary value for an investigation. More specifically, the possibility of full or partial reconstruction of a crime scene through network and/or device forensics of these was explored. Besides Android, other platforms hosting social media apps, such as BlackBerrys and iPhones, have also been studied with respect to their utilization in a digital forensics investigation (Al Mutawa et al., 2012; Tso et al., 2012).

Device and network forensics of popular apps have been carried out regularly. As a leading platform for secure voice, video and chat services, studies on Skype have become fundamental in this domain (Dupasquier et al., 2010; Molnár and Perényi, 2011). Similarly artifacts of Viber (Appelman et al., 2011; Marik et al., 2015), Telegram (Anglano et al., 2017) on Android smart phones, Telegram on Windows phone (Gregorio et al., 2017), WeChat (Wu et al., 2017), ChatSecure (Anglano et al., 2016), Wickr (Mehrotra and Mehtre, 2013), KiK on iOS (Ovens and Morison, 2016) and WhatsApp (Anglano, 2014; Karpisek et al., 2015; Majeed et al., 2015) have also been studied in detail (Azfar et al., 2016). contains a detailed research of Android forensics, and where 30 Android apps were studied with a focus on extracting useful information from memory using XRY - a well known mobile forensics tool. A generic taxonomy of the Android forensics is also proposed and which is correlated as a study of forensics artifacts of these apps.

Following the same motivation, we chose IMO apps for forensics study and which covered both device- and network-based analysis. For the device forensics part, we carried out a study of IMO files structure on Android and iOS platforms. This provided a way to find the critical artifacts and their significance. Similarly, a study of network forensics of IMO is presented on both the platforms in which network traffic is analyzed to classify the IMO flows and detection of user activities by analyzing the sniffed traffic. Novelty of our work lies in the study of IMO at such a scale. Moreover, our methodology of network forensics is interesting because most of the related research is limited to plaintext, whereas we have included encrypted network traffic in our study.

Contrary to other studies of chat and calling apps where network traffic is sniffed and then analyzed to draw important conclusions, a firewall is used to carry out analysis of IMO client-server communication in a controlled environment. Observing a known behavior of a IMO client, a firewall is then used to restrict the IMO client traffic and to force it to expose all the alternate connectivity methods. Our methodology of studying network traffic of IMO in a controlled environment with a firewall is generic and can be used for network forensics of other apps as well.

Referring to the goal of forensics analysis which should exhibit the specific properties listed in (Anglano et al., 2017), and where our methodology of both the domains of device and network forensics was aimed to achieve completeness, repeatability and generality. The limitations, we observed during the course of our study are clearly highlighted for the future work. It is important to mention that the results of this paper are presented with reference to specific versions of Android and iOS, but the conclusions are applicable to contemporary versions of Android and iOS with their supporting handsets.

Device forensics of IMO

Analysis methodology

We carried out device forensics of the IMO app on Android and on the iOS platform, separately. Our methodology of device forensics is focused to identify maximum possible artifacts from the device's internal and external memory. The analysis of IMO app on both the platforms of Android and iOS makes our work more comprehensive as each platform has unique identifiers and method of access to one are not applicable to the other.

The work flow of our methodology to carry out the device forensics of IMO is depicted in Fig. 1. It starts with assessing the functionalities of IMO app and which can be significant in generating artifacts that have good relevance in a forensics analysis. The impact of just installation of IMO app on the mobile is observed without configuring an account. In the next step, the account is configured and analysis of file structure of mobile memory/storage is carried out to identify any changes. All the important locations of

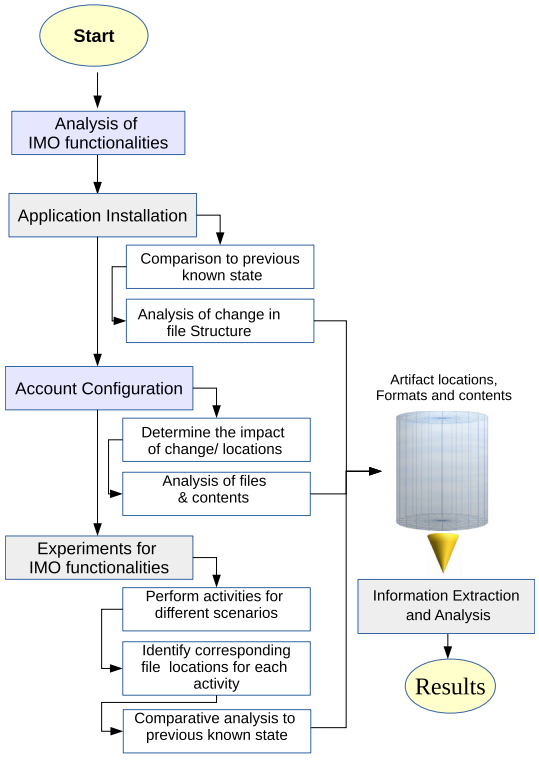


Fig. 1. Work flow of Device Forensics of IMO.

IMO files are then determined along with their formats. The last stage are experiments for functionality analysis at the start of a functional aspect and then analyze the changes in files of IMO. The files of IMO can be extracted out of mobile memory whenever required during any of the above for their content analysis and correlation. Artifacts are then studied in order to identify their mappings to different activities of an IMO user.

Analysis of IMO functionalities

As shown in Fig. 1, the first step of our work flow is to identify those functionalities of IMO app which are important for the forensics analysis of the mobile device. Like any calling and chat app, IMO provides a long list of features including live chat, voice call, video call, media share (Photos and video), story sharing, group chat, and so on. Starting from the IMO app installation in a mobile device, different app features and activities of user which leave possible traces of information in mobile storage are:

a) *Installation of IMO app.* Different chat and calling apps follow different file structures upon installation. Many files and folders are created in both user and system space during the installation process. Identifying the location of these files and folders along with their usage can be important for an investigator.
b) *Account configuration.* After the installation, the account configuration of IMO user is an important event and which has a unique impact on file storage. According to the specific user credentials and app permissions that are granted, a number of databases of IMO are updated.
c) *Fetching contacts list of users from a mobile.* Depending upon the settings and permissions, IMO fetches contacts which are already stored on the device. Details of these contacts and their format can help in forensics analysis of IMO.
d) *Exchange of chat messages.* IMO provides the functionality of message exchange including voice clips, stickers, images, videos, and so on. All chats are encrypted before transmission, however the IMO app stores these messages in plaintext form. The location and format of each type of these artifacts is important from forensics point of view.
e) *Status of chat messages.* Like other contemporary apps, IMO maintains the status of chat messages including delivered, read, and so on. Specific identifiers indicating the status of different messages are fixed by IMO.
f) *Voice and video calls.* IMO provides the functionality of audio and video calls which are encrypted. Records of these calls is maintained in the memory and their location could be important to any crime scene investigation involving IMO.

Results for Android and iOS forensics provided a good deal of similarity. However, to give clarity on access to mobile storage in each case, there are differences in file structures, and peculiar format of artifacts from the extraction from mobile memory/storage and thus Android and iOS will be outlined separately.

Device forensics on android

Experimental setup

We installed IMO on a rooted Samsung Galaxy 6.0 having an Android version of 6.0.1. On a Ubuntu 16.04 terminal, the following steps were taken to access the smart phone's file structure:

- Run the command "adb shell". This was to run the Android Debug Bridge (adb) tool in order to access the mobile memory.
- Pressed "Allow" on mobile against the prompt "Allow access to mobile data".
- User then entered to "shell@zeroflte:/$ ".
- To gain root previligaes, we pressed "su". Root privileges were shown as "root@zeroflte:/#".
- Entering the command "ls" will display all the files and folders on the mobile in both user and system space.

To identifying the file structure of IMO app, we used a number of controlled activities related to text, voice, and video chats, and observed their corresponding traces within different storage records maintained by IMO. These were validated using the Helium Backup (Chris, 2014 [accessed 3-june-2017]) to retrieve.db files of the IMO app. Finally, an Android backup extractor and db browser for SQLite was used to analyze the contents of storage elements against each activity.

Results and their analysis – android device

Following the steps of our work flow as shown in Fig. 1, the file structure of IMO on Android is identified and shown in Fig. 2.

The following section will throw some light on the files and folders that were identified during device forensics of IMO on Android platform.

Main folders of IMO file structure. Upon installation of IMO on Android, three main folders are created at fixed locations as:

- *com.imo.android.imoim-1* at location "/data/app/".
- *com.imo.android.imoim* at location "/data/data/".
- *IMO* at location "/data/media/0/". This is a location of SD card or user accessible memory and which is normally visible through USB connectivity on a PC, even without the root privileges.

The sub folders and files which are created within each of the main folders are depicted in Fig. 2. We now discuss the important files, their locations and related artifacts stored by IMO with their mapping to functionalities discussed in Section Analysis methodology.

File structure of com.imo.android.imoim-1. Like other popular chat and calling apps namely WhatsApp, Signal and Telegram, IMO creates this folder in device memory at "/data/app/". Each of these apps store their basic apk files along with other associated files in *app* folder. IMO also creates a folder with its name IMO in *app* folder and *base.apk* file is stored there.

File structure of com.imo.android.imoim. IMO stores most of its user related artifacts in this folder. This location is critical because artifacts related to user credentials/activities are stored there. We experimented number of user activities outlined in Section Analysis methodology and observed their corresponding storage patterns by IMO. Referring to Fig. 2 again, the following explains the files and folders at this location.

a) ***"cache" folder***. IMO stores sent images and videos in a "cache" folder, whereas received images and videos are stored in user space within *IMO* folder. This will be discussed in detail in a following section. All these images and videos are stored in non-encrypted form in the device memory.
b) ***"Databases" folder***. From forensics point of view, *Databases* folder is considered as one of the most important component of file structure of IMO. Upon installation of IMO app, the following two files are created in this folder before the account configuration:
   - imofriends.db.
   - imofriends.db-journal.
c) After the user account is configured with IMO servers, the app creates two additional files in *Databases* folder:

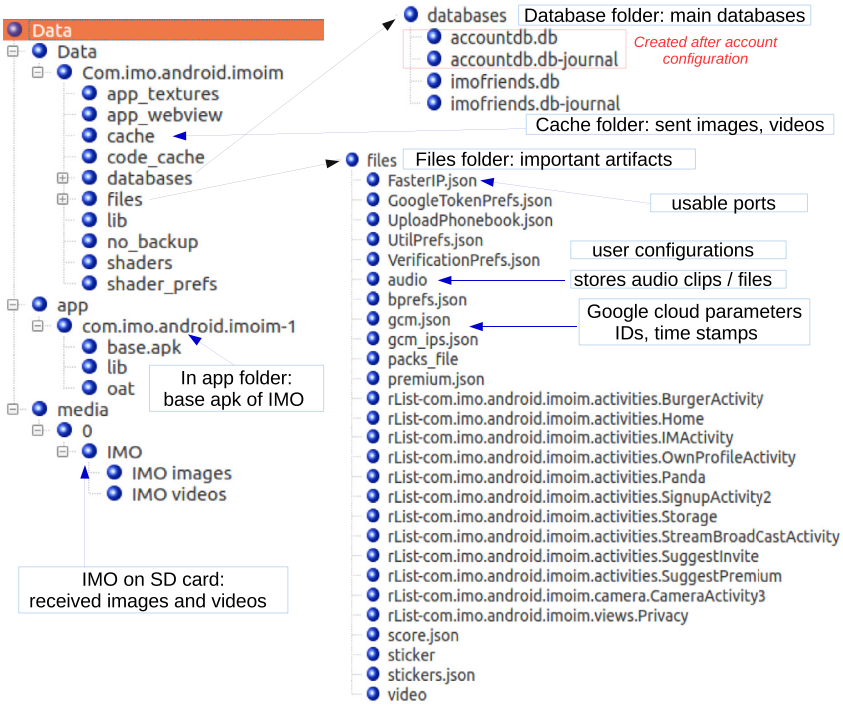


Fig. 2. File structure of IMO in Android.

- accountdb.db.
- accountdb.db-journal.

imofriends.db is an important folder as it contains wealth of information related to the contacts, their chats, timestamps, telephone numbers, emails, and a great deal of other information.

imofriends.db. This is the main database file in which IMO maintains the records of users, and their associated activities in a plaintext form. These records include number of tables containing information of evidentiary value, such as phonebook entries, email IDs, call logs, chat logs and chat messages, and so on. Different tables created by IMO in this location are shown in Fig. 3. We observed plaintext data in these tables. The data is relates to activities including chats and call logs as defined in Section Analysis methodology. We now discuss the contents of these tables along with their mapping to the user activities, as these can help in crime scene reconstruction.

i) *imo_phonebook.* During the installation process, IMO accesses the mobile phonebook data according to the permissions granted and saves them in the table 'imo_phonebook'. The structure of this table is shown in Fig. 4. We observed that IMO stores all the critical information related to the contacts in plaintext form as shown in Fig. 5. We carried out experiments on number of user accounts and verified that against any unique mobile number of a user, IMO assigns a unique identifier naming "uid" and which is fixed. Any IMO user can be tracked through this unique identifier and is therefore can be considered to be one of most important artifacts of IMO. Moreover. IMO also keeps the record of the number of incoming and outgoing GSM calls and GSM messages (SMS) in this table. The reasons for storing this information about calls and SMS is unknown, but it can prove helpful information with an investigation.

ii) A separate record of all users and their mobile numbers is stored in the table 'phonebook_numbers' as shown in Fig. 6.

iii) A similarly table 'messages' is used to store chat messages in plain text as shown in Fig. 8. It is important to note that even the email addresses and phone numbers of the IMO users are stored in the column 'imdata' of this table in plaintext. Through repeated experiments of chat messages for different scenarios of text, images, videos, and so on, we decoded values of different fields of the Table as shown in Fig. 7. It can be useful to forensics analysis of IMO during any investigation.

iv) IMO also tracks timestamps of different activities which are stored in the tables discussed above. Through forced activities of messaging on IMO, we verified our results of time stamps using on-line conversion tool ("http://timestamp.online/").

v) Table 'calls_only' stores call logs in plaintext as shown in Fig. 9.

vi) Table 'stories' is used to save artifacts related to *Story sharing* feature of IMO as shown in Fig. 10. All these details are stored in plaintext form.

vii) *Contents accessible from IMO servers.* One of the most critical result of our study of IMO is on-line accessibility of uploaded contents (images and videos) from its servers without any authentication. A field of *object_Id* exists in different tables within *imofriends.db* and which contains the direct link to a specific uploaded content at IMO servers. By just entering this *object_Id* in a browser as URL (https://imo.im/s/object/object_Id/), content uploaded by IMO client is accessible directly. During an investigation, retrieving the *object_Id* from *imofriends.db* will enable an investigator to retrieve the exact content shared by the IMO servers, even if the contents are deleted from the mobile device. Fig. 11 shows that different locations of *object_Id* in tables within *imofriends.db*. *object_Id* of uploaded contents are either placed in the *imdata* record or in the *icon* record for profile images in different tables.

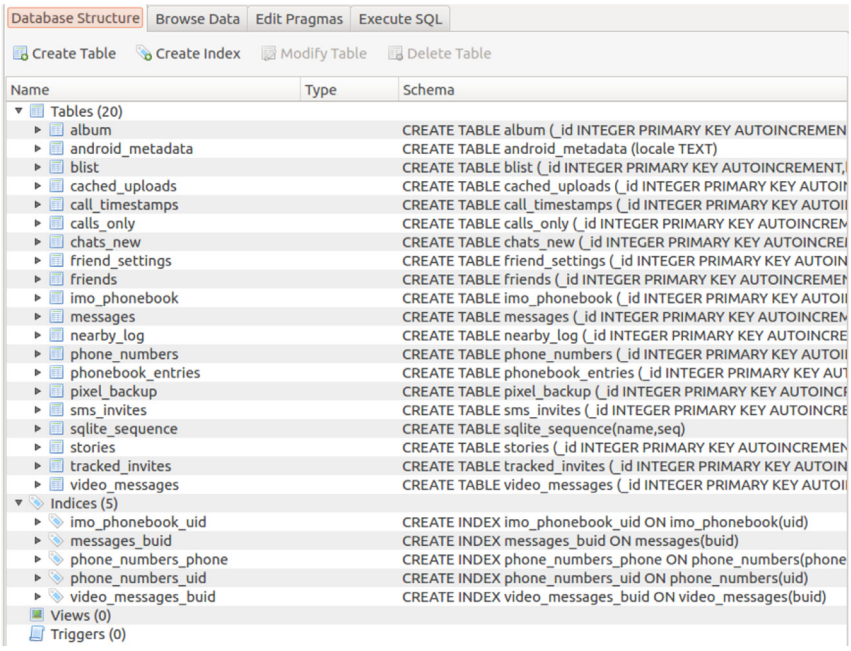


Fig. 3. DB tables of IMO

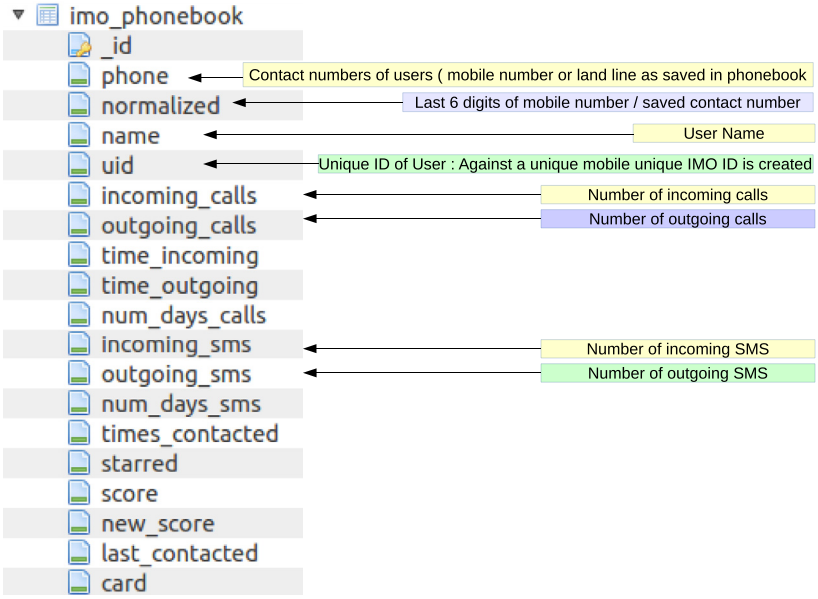


Fig. 4. Fields of imo_phonebook table.

d) ***Files folder.*** *Files* is another important folder in which number of artifacts are sorted by IMO and which has forensics value. The detailed view of contents within *Files* folder has been marked in Fig. 2. Through repeated experiments of user activities and analysis of files within *Files* folder, the following files are considered significant having important contents:

i) *FasterIP.json.* This is JSON file which contains the useable TCP ports of IMO. FasterIP.json shows that IMO designates the ports of 5228, 5223 and 443 for client to server connection establishment. Though device and network forensics of IMO were carried out as independent studies, results of one portion support the findings of the other. Our network analysis part also highlights the IMO connectivity patterns on useable ports of 5228 and 5223. If these ports are blocked, IMO still communicates on 443 which is a standard port of HTTPS. This unique feature of IMO connectivity pattern makes it a difficult choice for app blocking solutions. We will explain this design flexibility of IMO with more details in Section Network forensics of IMO later.

ii) *audio folder.* IMO stores audio files, recordings and clips in *audio* folder. By repeating number of forced activities on IMO client, we verified that IMO always stored all audio clips/files in plaintext form at this location.

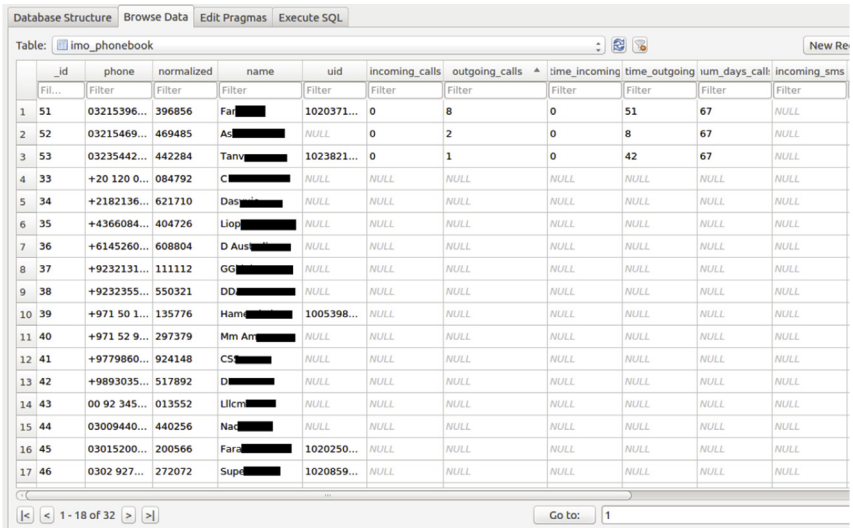


Fig. 5. IMO phonebook

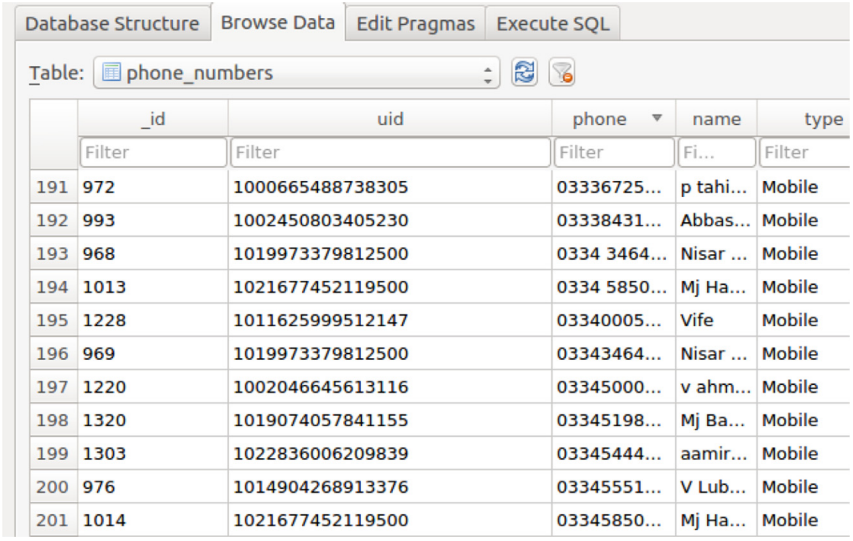


Fig. 6. Record of phone numbers.

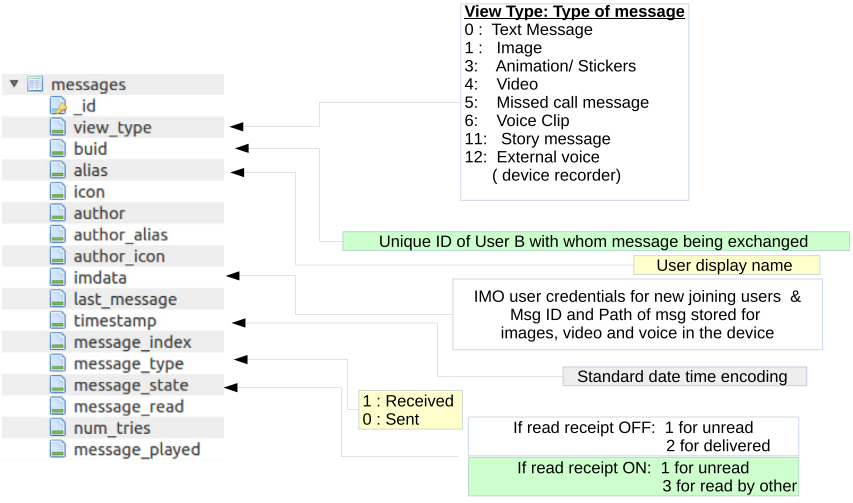


Fig. 7. Record of IMO messages.

Fig. 8. Messages table of IMO.

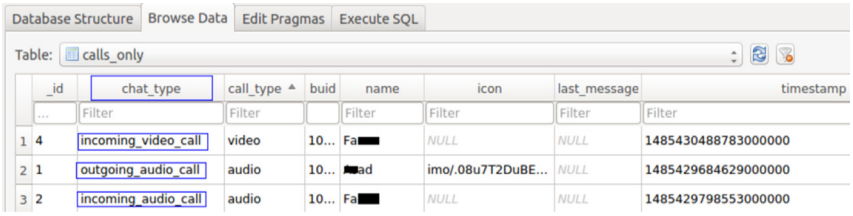


Fig. 9. Calls table of IMO.

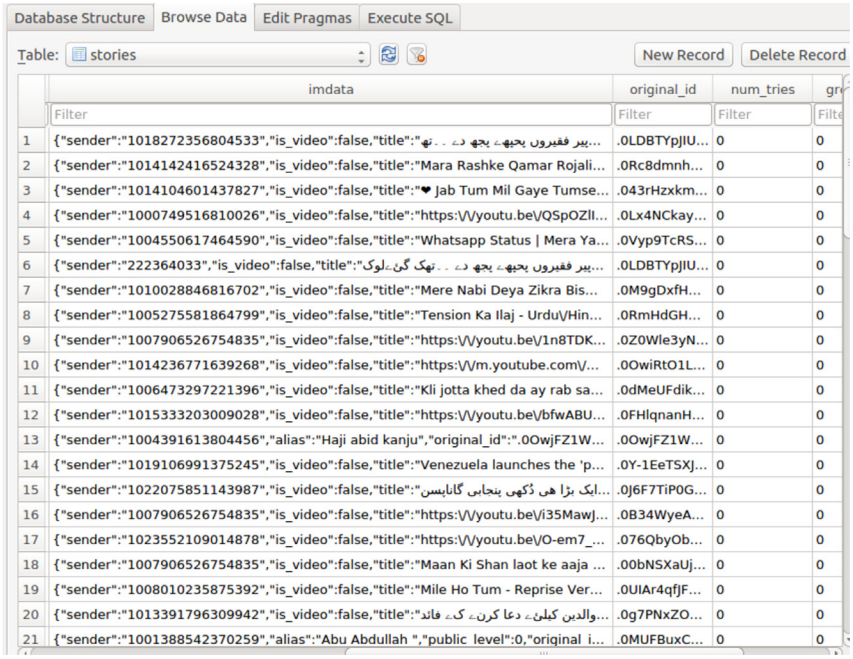


Fig. 10. Location of story sharing by IMO in memory.

iii) *video folder.* During entire period of our study, video files were observed to be stored in either user accessible memory in the case of received files or a *cache* folder in the case of sent files in a plaintext form. The *video* folder within *Files* folder was observed to remain empty for all possible scenarios of video sharing.

File structure of IMO. IMO maintains a folder named *IMO* in user accessible memory of the device. Within this folder, IMO maintains two sub folders of *IMO images* and *IMO videos*. These are for storing received images and videos respectively in user accessible memory, as shown in Fig. 2. Like all other artifacts in the device, IMO also stores the received images and videos in non-encrypted form.

Device forensics of IMO on iOS

Experimental setup

For our study of IMO artifacts on the iOS system, we used a jail-broken Apple iPhone 5 with iOS 9.3.3, along with Cydia and Apple File Conduit 2 (AFC2) (Chris, 2014 [accessed 3-june-2017]) in order to access the files of interest. A step-wise methodology to gain access of IMO storage location and information extraction is:

a) We installed Cydia® in our iPhone and gained access to its storage in following steps.
   i) Installed the secure shell "Openssh package" on the iPhone.
   ii) From WiFi settings, we noted the IP address assigned to the iPhone.

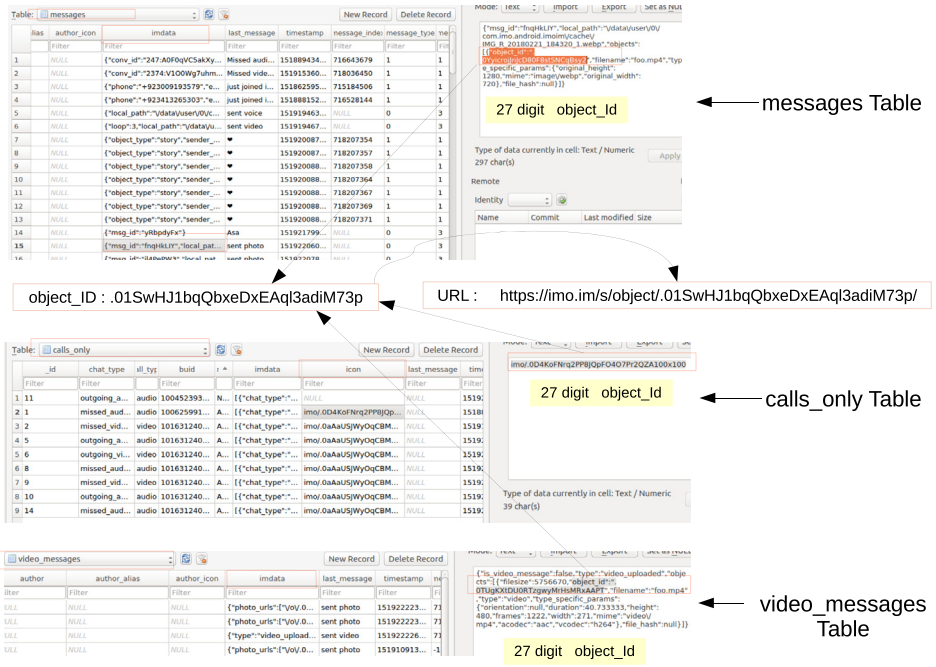


Fig. 11. Contents at IMO servers.

iii) Opened a terminal in our Ubuntu 16.04 host. The same can be attained on a Microsoft Windows platform through the use of PuTTY.(http://www.putty.org/).
iv) Run the command "ssh root@[IP address of iPhone]" in terminal to gain the secure access of iPhone.
v) Entered the Password "alpine".
vi) The connection to the iPhone was successfully established after another prompt of Yes/No at the terminal. We were then connected to the iPhone as iPhone: ~ root #.

b) Similar to Section Analysis methodology, artifacts of IMO on iOS are also identified and discussed in the following section. To avoid any unnecessary repetition, we will explain only those aspects in detail which are different in case of iOS. Moreover, the similarities will also be briefly discussed.

Results and their analysis — iOS device

IMO app maintains different file structure in iOS as compared to what we have seen in case of Android. Analysis of iOS internal memory revealed file structure of IMO app as shown in Fig. 12.

Salients aspects of device forensics for iOS are as under:

a) Similar to Android, IMO stores all the artifacts on iOS device in plaintext form. It includes account information, friends list, phonebook, chat messages, audio clips, video clips, story sharing, and so on.
b) Main installer directory of IMO app on iOS is at the path:/*private*/*var*/*mobile*/*Applications*/1391315*F* – *D*105 – 40*BF* – *BCE*3 – 5371*F*278*E*603.
c) Within *Applications* folder, iOS creates folder for different installed apps with 32 character alpha-numeric names as unique identifiers (UID) as shown in Fig. 12.
d) We observed that UID name of IMO app changed on each installation.
e) IMO stores profile picture and shared pictures/images at:1391315*F* – *D*105 – 40*BF* – *BCE*3 – 5371*F*278*E*603/*Library*/*Caches*/*default*/*com.hackemist.SDWebImageCache.default*.
f) Similar to Android, IMO stores images in *WebP* format.
g) Path for captured images was identified as:1391315*F* – *D*105 – 40*BF* – *BCE*3 – 5371*F*278*E*603/*Library*/*Caches*/*libcache*/*com.hackemist.SDWebImageCache.libcache*.
h) Videos captured from the iPhone, but not sent, were found at:1391315*F* – *D*105 – 40*BF* – *BCE*3 – 5371*F*278*E*603/*tmp*/*imotmp*.
i) Sent and received videos or audio messages were identified at:1391315*F* – *D*105 – 40*BF* – *BCE*3 – 5371*F*278*E*603/*Library*/*Caches*/*videos*.
j) IMO stores user account credentials, phonebook contacts, chat messages, call records, and so on, in main database file (*IMODb2.sqlite*) at path:1391315*F* – *D*105 – 40*BF* – *BCE*3 – 5371*F*278*E*603/*Documents*/.
k) Similar to Android, IMO on iOS maintains records of chats and contacts in number of tables as shown in Fig. 13. In the case of iOS, a number of tables within *IMODb2.sqlite* are reduced to six, as compared to Android version of IMO where 20 tables were used. We observed that the number of records maintained within each table are also different for IMO on iOS.
l) The naming convention for the records in iOS is also different. It can be easily correlated to the set of activities found in Android, as shown in Fig. 14. It is a representation of a single case of *message* table for both Android and iOS platforms.
m) Besides storing all the contents of IMO app in a specific folder with UID number at path/*private*/*var*/*mobile*/*Applications*/, IMO also creates a snap folder at path /*home*/*snap*. within *snap*, another snap folder contains the same list of folders or

Fig. 12. File structure of IMO in iOS.

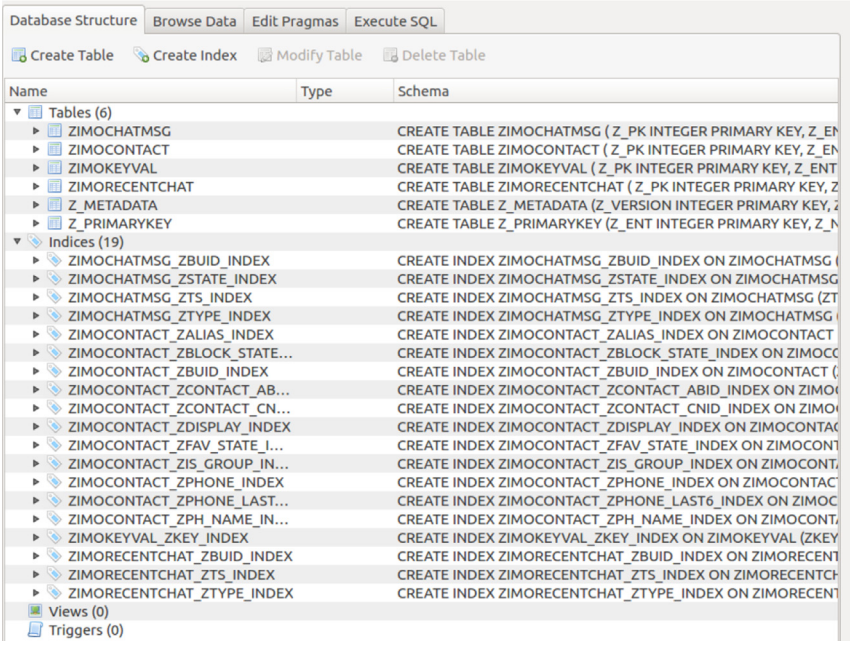


Fig. 13. Database structure of IMO in iOS.

files. The reason of this duplication is unknown to the authors.

Network forensics of IMO

Network forensics of IMO is the second dimension of our study. Extensive traffic analysis has been carried out to identify the particular app as a first step. Then the classification of user activities is done by finding certain fixed patterns in server-client sessions. As most of the chat and calling apps are secure and traffic flows are HTTPS encapsulated, access to actual contents of information being exchanged between an app client and the servers is difficult. However, identification of a particular app and its user's activities is made possible by establishing behavior analysis of the traffic.

We carried out extensive traffic analysis of IMO and found out number of fixed patterns which are considered useful to identify the IMO app over the network, and to classify user's activities. Similar to the device forensics part, we carried out study of network forensics of IMO for both the Android and iOS platforms. The results of our study are presented separately. We employed a unique model of traffic sniffing environment using a firewall which is described in subsection Experimental setup for network forensics.

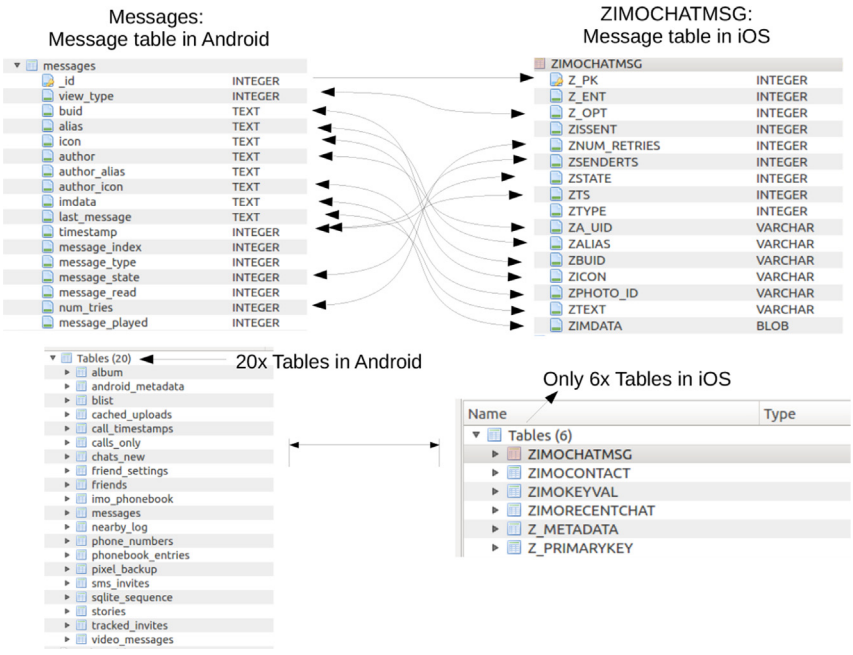


Fig. 14. Difference of records in IMO tables: Android vs iOS.

Experimental setup for network forensics

In order to intercept the network traffic, we established a network infrastructure as defined in Fig. 15. The target mobile device was connected to the Internet through a wireless access point, and all of the Internet traffic of wireless access point was routed through a firewall running on an open source customized distribution of FreeBSD and pfsense (Rubicon Communications, 2004–17 [accessed 3-june-2017]).

The firewall was configured to filter out Internet traffic and thus help us to focus only on the IMO traffic in a controlled environment. The network side of firewall traffic was mirrored on Layer 3 switch for live interception. This mirrored traffic was fed to our analysis setup. Our analysis setup comprised of widely-used network protocol analyzer Wireshark (www.wireshark.org) and a firewall. Based on the findings of traffic flows, the rule set of the firewall was updated after each experiment and to narrow down our focus to only packets of interest, and to reveal the secret connectivity patterns of IMO.

Why we used firewall?

In a normal network forensics studies, traffic is first sniffed and is fed to the analysis setup for detailed examination. To the best of our knowledge, the concept of incorporating firewalls in this type of a study has not been previously used for exploring the required artifacts from the traffic of social media apps. The motivation for employing firewall in traffic analysis of IMO is driven from a fact that developers of secure chat and calling apps to keep the flexibility of client to server connectivity in order to ensure the availability of different services. The reasons of service non-availability can be many including traffic monitoring/blocking solutions deployed at network nodes or other causes of network failures. As a

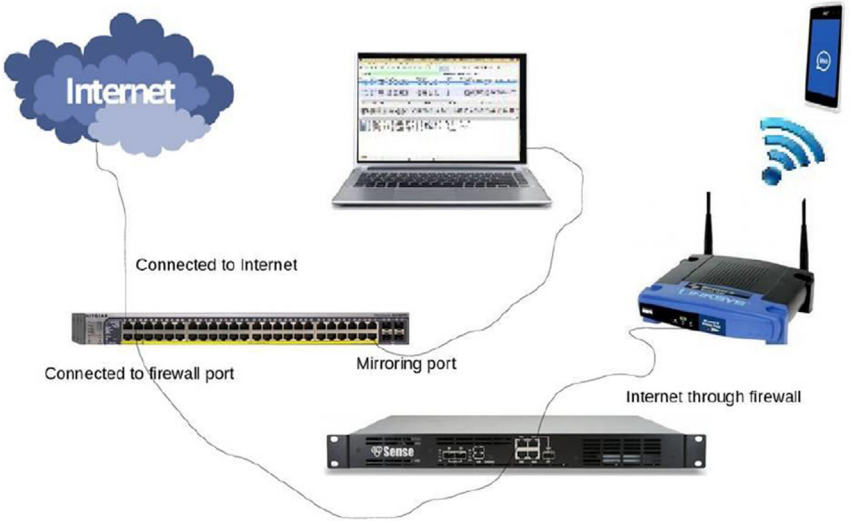


Fig. 15. Setup for traffic capturing.

result, a cluster of servers are often deployed and which are physically distributed to provide an alternate mechanisms for connectivity. This supports redundancy and anonymity. Server ranges and an alternate set of TCP/UDP ports and series of other parameters are often kept hidden to ensure availability of service. Thus, the problem domain of carrying out study of network forensics of IMO is characterized by following fundamental facts:

- The communication protocol between IMO client to its servers is not known.
- All sessions between IMO client and IMO servers are encrypted.
- Alternate mechanisms of connectivity are not known.

With the traffic available over the network, we propose the use of firewall in studying the encrypted traffic of IMO and reach from unknown to maximum possible known. The process is progressive in nature. Sniffing the traffic as shown in Fig. 15 with no restriction on firewall and its analysis lead us to the first stage observations, and which helped to define basic restrictions of TCP ports through firewall. Against the blocking on firewall, IMO changed its connectivity on other TCP ports. The firewall was then reconfigured with updated rule sets and this process continued to identify all the methods of connectivity of IMO client to its servers which are kept secret in the design. This helped us to develop correct model of detection of IMO and various events of user activities. Though our results are specific to IMO, the methodology of configuration of firewall according to the observed events of traffic analysis can be employed to study any app/service that exchanges data over the Internet. The role of firewall and type of different rule sets, which we employed to identify the IMO connectivity patterns, will be explained in more detail in Section Results and their analysis.

It is important to note that any social media app which is designed to provide privacy has to follow certain set of rules to enable itself to communicate over the TCP/IP stack. Even if the contents are secured, communication to the servers and their responses often leaks sensitive information. We studied these leakages for thousands of samples and important deductions are made about the parties involved in communication, including the type of information they are exchanging like text chats, voice messages, video chats and file sharing.

Though the structure of IMO communication protocol and security architecture was unknown to us, we analyzed its traffic dumps against the possible list of user activities (described in Section Identification of IMO traffic). The known knowledge of OSI and TCP/IP communication layers was mapped to the filtered traffic, and which was fed to the analysis setup to draw certain signatures of encrypted traffic (described in Section Identification of IMO traffic). To bring out important artifacts of IMO. We thus used a combination of techniques such as port-based analysis, inspection of bytes exchanged between the client and server, and a behavior analysis in a controlled network environment. The use of firewall in this study helped us to identify alternate connectivity mechanisms of IMO and which usually remains obscured in normal traffic sniffing scenarios.

Results and their analysis

IMO supports encrypted traffic for chats, voice and video calls. From these encrypted sessions, we were not able to reconstruct them to retrieve any helpful information. However, we were able to study important artifacts from extensive traffic analysis of IMO in relation to the series of forced activities. The list of forced activities is described in Section Identification of IMO traffic.

Although we have only studied IMO, the methodology in Section Experimental setup for network forensics is generic and can be used to study other related applications. During this study, we have engineered the role of firewall in a network traffic sniffing environment to clearly determine the unknown communication protocol associated with IMO. Traffic was analyzed for a series of triggered events of messaging, voice and video calls, in order to define consistent patterns through filtration. Based on these patterns, different rule sets were configured on firewall to filter out the specific traffic. This phenomenon not only helped us stop or filter the targeted traffic, but also helped us to discover the network connectivity patterns used by IMO. Our results indicate that this flexibility could prove to be a problem area for law enforcement agencies in blocking IMO in some countries of the world.

TCP ports

Like many other eXtensible Messaging and Presence Protocol (XMPP) based chat applications (GitHub, 2017 [accessed 3-june-2017]), IMO uses the standard TCP ports of 5222 and 5223 (XMPP over SSL). The use of firewall helped us to identify hidden design flexibilities of IMO connectivity and which provides the reason as to why IMO is hard to block. Table 1 shows a case scenario of using firewall to enforce strict control on IMO traffic through TCP ports only and progressive re-configuration of firewall after each observation of IMO client connectivity to its servers (see Table 2).

We also observed a good deal of IMO chat sessions using TCP on port 5228. Once we blocked TCP connection on Port 5228 for IMO client to its server through firewall, IMO established connections on 5222 and then on 5223. Once we updated the firewall rule sets to block all these ports, IMO continued to exchange messages on TCP port 443 (HTTPS). It is important to note that port 443 is used for all HTTPS traffic and blocking 443 can have undesired consequences, and is this is perhaps one of the hurdles in blocking IMO. We further observed that the communication on the aforementioned ports could be chat, voice/video, adding/removing a contact, or any other messenger activity. Thus TCP ports of 5222, 5223, 5228 and 443 could be observed over the network for IMO sessions.

Server ranges and traffic behavior

IMO uses SSL/TLS to encrypt voice, video and chat conversations, and where the client is designed to initiate connections with number of servers depending upon a few unknown conditions. We added these IP subnets in firewall rules and forced IMO client to switch over to alternate connection options with its servers. In a few cases, we observed that the IMO client established the connection with two servers initially, and then communicated through one of these servers, while maintaining the connection in parallel with the other server. Fig. 16 clearly shows that two parallel connections are established by IMO client with its servers at 192.12.31.89 and 108.177.15.188. Network forensics helped us to discover the IMO connectivity patterns. For this we identified the server ranges being used by IMO in our geographical region, and progressively blocked the identified server ranges on the firewall and observed that IMO was still able to communicate. Server ranges of 64.233.0.0/16, 207.154.0.0/16, 74.125.0.0/16, 108.177.15.0/24 and 192.12.31.0/24 were observed for IMO connectivity.

Servers with IP range of 192.12.31.0/24 are only designated for IMO publicly. Filtering this range through the firewall forces the IMO client to switch over to Google's servers at 64.233.187.0/24 for its services. This behavior of IMO was noted repeatedly for number of call and chat events. In normal connectivity scenarios, once the firewall was configured on an all pass rule set, the IMO client always connected with one of the servers in the range of 192.12.31.0/24.

Table 1
Firewall rules for TCP ports used by IMO.

Step	Protocol	Source Port	Dest Port	Action	Observation
I	Any	Any	Any	Any	Server TCP port connections on 5228
II	TCP	5228	Any	Block	TCP port connections on 5222
	TCP	Any	5228	Block	
III	TCP	5228, 5222	Any	Block	TCP port connection on 5223
	TCP	Any	5228, 5222	Block	
IV	TCP	5228, 5222, 5223	Any	Block	TCP port connection on 443
	UDP	Any	5228, 5222, 5223	Block	
V	TCP	5228, 5222, 5223, 443	Any	Block	TCP connection failed
	TCP	Any	5228, 5222, 5223, 443	Block	

Table 2
Traffic characteristics of IMO on Android.

Event	Description
Access to IMO messenger application	As explained earlier
User A is typing	• Client **Beginning:** packets with 160 bytes of payload from client to server **Incremental:** 16 bytes depending on message length • Server 8 bytes of payload sent in acknowledgement of each packet sent from the target
User B is typing	• Server **Beginning:** packets with 144 bytes of payload from Server to Target **Incremental:** 16 bytes depending on message length • Client 64 bytes of payload sent in acknowledgement of each packet sent from the server
Call initiated from User A	• Client Two consecutive packet having payload size of 272 bytes and 128 bytes respectively are sent to the server • Server In response of these messages, the server acknowledges with 48 bytes of packets. In less than a second, a series of packets are also sent by the server in which one of the packet size is greater than 1000 bytes and the others are short in length.
Call initiated from User B	• Server Packet having payload size of 112 bytes is received from the server • Client Target mobile acknowledges by sending a packet of payload size 64 bytes • Server In less than a second, a series of packets are also sent by the server in which one of the packet size is greater than 1000 bytes and the others are short in length • Client Two consecutive packet having payload size of 64 bytes and 176 bytes respectively are sent to the server
Voice/video call termination	UDP stream flowing from the target IP stops

No.	Time	Source	Destination	Protocol	Length	Info
4220	454.253498	192.168.15.105	108.177.15.188	TCP	66	34857 → 5228 [ACK] Seq=3689 Ack=
4219	454.250788	192.168.15.105	192.12.31.89	TCP	66	38762 → 5228 [ACK] Seq=21601 Ack
4218	454.250061	108.177.15.188	192.168.15.105	TCP	280	5228 → 34857 [PSH, ACK] Seq=1484
4217	454.250060	108.177.15.188	192.168.15.105	TCP	1414	5228 → 34857 [PSH, ACK] Seq=1350
4216	454.249772	108.177.15.188	192.168.15.105	TCP	609	5228 → 34857 [PSH, ACK] Seq=1295
4215	454.199881	192.12.31.89	192.168.15.105	TCP	94	5228 → 38762 [PSH, ACK] Seq=2131
4214	454.199880	192.12.31.89	192.168.15.105	TCP	1414	5228 → 38762 [ACK] Seq=19969 Ack
4213	454.154165	192.12.31.89	192.168.15.105	TCP	178	5228 → 38762 [PSH, ACK] Seq=1985
4211	453.824168	192.12.31.89	192.168.15.105	TCP	66	5228 → 38762 [ACK] Seq=19857 Ack
4210	453.467595	192.168.15.105	192.12.31.89	TCP	130	38762 → 5228 [PSH, ACK] Seq=2153
4209	453.438017	192.168.15.105	108.177.15.188	TCP	66	34857 → 5228 [ACK] Seq=3689 Ack=
4208	453.437778	192.168.15.105	192.12.31.89	TCP	66	38762 → 5228 [ACK] Seq=21537 Ack
4207	453.284528	108.177.15.188	192.168.15.105	TCP	273	5228 → 34857 [PSH, ACK] Seq=1275
4206	453.254341	192.12.31.89	192.168.15.105	TCP	178	5228 → 38762 [PSH, ACK] Seq=1974

Fig. 16. Two TCP connections on port 5228.

This shows that up to 255 servers are dedicated for IMO services, but we observed that it actually doesn't always happen. Moreover, any Google server hosting the IMO might be assigned to provide the services to users in different conditions of connectivity.

From the observed behavior of IMO client-server connectivity with alternate connections, we can safely make following inferences:

a) The alternate connection is used to initially check the reliable connectivity and then to balance the traffic load among the servers.
b) This alternate connection acts as a backup for the communication because it provides the same services to the client as any actual IMO server.

c) IMO transfers its activities to the Google servers, hosting IMO only, during the prolonged user activities in order to offload the traffic on the designated IMO servers.
d) The connection patterns of this alternate connection are the same as of the other IMO servers.

The traffic samples collected between the IMO client and all these server ranges were separately analyzed. From network forensics perspective, we found IMO to be a real challenge. Extracting traffic behavior from encrypted contents is itself a difficult paradigm. A further challenge is added because of the random behavior of server-client connectivity. However, through a special scenario of our traffic collection and canalizing the IMO client to connect to its servers in a controlled environment set out by our firewall, we were finally able to detect patterns of IMO traffic for server ranges of 192.12.31.0/24. We focused on representing the characteristics of IMO traffic that could be helpful to identify different events of IMO messenger.

Identification of IMO traffic

We collected a large number of traffic samples by using the Android and iOS mobile devices separately. To filter out the IMO communication between the target mobile and the server as shown in Fig. 15, we defined following activities against which we tried to identify specific signatures from encrypted communications.

a) Access to IMO messenger application; when a user clicks to open it on the client device.
b) Target mobile (User A) is typing. Target mobile is the mobile under observation and being used by User A, while the other mobile is the one with whom User A is communicating and thus identified as User B.
c) User B is Typing.
d) Call initiated from User A.
e) Call initiated from User B.
f) Voice/video call termination.
g) User A sends the message to User B which is still unread.
h) Message is seen as read at chat thread of User A.
i) Message not delivered to User B yet (User B is off-line).
j) Story sharing.
k) Group messaging.

After carrying out extensive traffic analysis of encrypted sessions between IMO client to its server, we could identify certain fixed patterns against first six activities only. For other listed activities, behavior of IMO is quiet random for different connectivity scenarios and identification of specific signatures could not be made with accuracy.

Against the first six listed activities of IMO, let us explain the results of network forensics in detail. As the behavior of encrypted IMO traffic on both the platforms is different, we will cover them separately in the succeeding paragraphs.

Traffic characteristics on android. We will demonstrate the mechanism used for identification of above events of IMO for Android case firstly. The IP address of client was 192.168.15.105 and IP range of IMO servers considered during the study was 192.12.31.0/24.

Access to IMO messenger application. After capturing the network traffic of Android running on Samsung Galaxy 6 mobile, our first objective was to gain access to IMO traffic explicitly for further analysis. We had identified the ports on which IMO communicates (Section TCP ports). On the captured traffic dump, we applied the filter on each of these TCP ports and found IMO traffic on port 5228 as shown in Fig. 16.

In 90% of the scenarios during the entire period of our study, we observed that IMO client established two parallel TCP connections, at the same time, with its servers. One of the TCP connections was always with the dedicated IMO server sitting in the range specified in Section Server ranges and traffic behavior and the other (the alternate connection) could be with any of the Google servers hosting IMO. We observed that the connection with the alternate server was continuously maintained in parallel while communicating with actual IMO server. It was also observed that IMO transferred its activities to the Google servers instead of its own designated servers during prolonged user activities through IMO client. The use of this alternate connection was revealed at a number of instances, and we assume that this behavior is perhaps for load balancing and to create a reliable connectivity.

It was also discovered that extra services were provided when the user was connected with the dedicated IMO server. For instance, whatever User A was typing it was also being shown on the screen of User B, even before pressing the send button during a text chat session. The message remained on the screen of User B, if User A pressed send button. However, the message disappeared from the screen of User B, if the connection between the communicating parties was lost before pressing the send button. This phenomenon was not observed on sessions established through alternate Google servers.

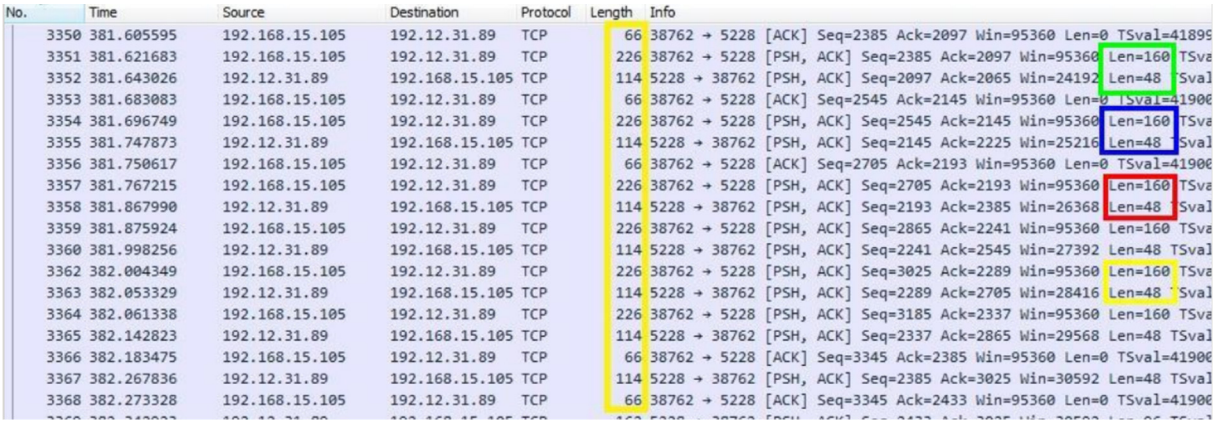
Maintaining alternate connectivity through different Google servers (e.g. 64.233.187.0/24) and common HTTPs ports (such as port 443) is a unique design approach of IMO which makes its services uninterrupted, even in the countries where other social media apps or VPNs are blocked.

In Fig. 16, the IP address of the target mobile is 192.168.15.105. It established one connection with 192.12.31.89 (IMO server) on port 5228 and a parallel alternate connection with 108.177.15.188 (Google Server hosting IMO) on port 5228. The connection with IP address 108.177.15.188 became idle after an exchange of a few packets. So, we filtered out data streams between 192.168.15.105 (client) and 192.12.31.89 (IMO server) for further analysis and detection of user activities on IMO.

User A is typing. During the chat conversations, the event of typing by User A was also investigated by tracking the flow controls of communication and packet analysis. We observed a specific pattern of TCP packets with fixed payload size over the network when User A was chatting with User B as shown in Fig. 17. We tested these results on hundreds of traffic samples and finalized our conclusions.

At the beginning of a chat activity, packets with a fixed TCP payload of 160 bytes were transferred from User A to IMO server and this TCP pattern continued till the message was actually sent by the User A. In the acknowledgement of each TCP packet sent by User A to the IMO server, a packet with a payload size of 48 bytes was received from the IMO server. The activity of typing by User A of IMO chat conversation can be easily identified from the network traffic by using the TCP patterns discussed above.

Let us explain our conclusions in relation to Fig. 17 in more detail. TCP is a reliable connection oriented protocol which supports sequence and acknowledgement numbers to guarantee successful transfer of data between the communicating parties. So, in every TCP connection, any party either sends its data along with the acknowledgement of the data received by the other party or sends only the acknowledgement with the empty payload if no data is to be sent. As in chat conversation, both the parties can communicate simultaneously, so any one can send the data along with the acknowledgement of the data (amount of bytes) received by the other party at that particular time and if it receives the data after that then it has to send the acknowledgement of this data received



No.	Time	Source	Destination	Protocol	Length	Info
3350	381.605595	192.168.15.105	192.12.31.89	TCP	66	38762 → 5228 [ACK] Seq=2385 Ack=2097 Win=95360 Len=0 TSval=41895
3351	381.621683	192.168.15.105	192.12.31.89	TCP	226	38762 → 5228 [PSH, ACK] Seq=2385 Ack=2097 Win=95360 Len=160 TSva
3352	381.643026	192.12.31.89	192.168.15.105	TCP	114	5228 → 38762 [PSH, ACK] Seq=2097 Ack=2065 Win=24192 Len=48 TSval
3353	381.683083	192.168.15.105	192.12.31.89	TCP	66	38762 → 5228 [ACK] Seq=2545 Ack=2145 Win=95360 Len=0 TSval=41900
3354	381.696749	192.168.15.105	192.12.31.89	TCP	226	38762 → 5228 [PSH, ACK] Seq=2545 Ack=2145 Win=95360 Len=160 TSva
3355	381.747873	192.12.31.89	192.168.15.105	TCP	114	5228 → 38762 [PSH, ACK] Seq=2145 Ack=2225 Win=25216 Len=48 TSval
3356	381.750617	192.168.15.105	192.12.31.89	TCP	66	38762 → 5228 [ACK] Seq=2705 Ack=2193 Win=95360 Len=0 TSval=41900
3357	381.767215	192.168.15.105	192.12.31.89	TCP	226	38762 → 5228 [PSH, ACK] Seq=2705 Ack=2193 Win=95360 Len=160 TSva
3358	381.867990	192.12.31.89	192.168.15.105	TCP	114	5228 → 38762 [PSH, ACK] Seq=2193 Ack=2385 Win=26368 Len=48 TSval
3359	381.875924	192.168.15.105	192.12.31.89	TCP	226	38762 → 5228 [PSH, ACK] Seq=2865 Ack=2241 Win=95360 Len=160 TSva
3360	381.998256	192.12.31.89	192.168.15.105	TCP	114	5228 → 38762 [PSH, ACK] Seq=2241 Ack=2545 Win=27392 Len=48 TSval
3362	382.004349	192.168.15.105	192.12.31.89	TCP	226	38762 → 5228 [PSH, ACK] Seq=3025 Ack=2289 Win=95360 Len=160 TSva
3363	382.053329	192.12.31.89	192.168.15.105	TCP	114	5228 → 38762 [PSH, ACK] Seq=2289 Ack=2705 Win=28416 Len=48 TSval
3364	382.061338	192.168.15.105	192.12.31.89	TCP	226	38762 → 5228 [PSH, ACK] Seq=3185 Ack=2337 Win=95360 Len=160 TSva
3365	382.142823	192.12.31.89	192.168.15.105	TCP	114	5228 → 38762 [PSH, ACK] Seq=2337 Ack=2865 Win=29568 Len=48 TSval
3366	382.183475	192.168.15.105	192.12.31.89	TCP	66	38762 → 5228 [ACK] Seq=3345 Ack=2385 Win=95360 Len=0 TSval=41900
3367	382.267836	192.12.31.89	192.168.15.105	TCP	114	5228 → 38762 [PSH, ACK] Seq=2385 Ack=3025 Win=30592 Len=48 TSval
3368	382.273328	192.168.15.105	192.12.31.89	TCP	66	38762 → 5228 [ACK] Seq=3345 Ack=2433 Win=95360 Len=0 TSval=41900

Fig. 17. User A typing – android.

in a separate packet with empty payload. So, ID [3350], [3353] and [3356] in Fig. 17 represents the acknowledgement of every 48 bytes received by the server with the empty payload (Len = 0).

A scenario can arise that while User B is not having a connection to its server, and User A is sending the data. Traffic pattern described above will remain the same in this case also. As IMO chat conversations are done through IMO server not with the other IMO user directly, User A will be communicating with the IMO server and sending/receiving the packets with a payload size of Len = 160 and Len = 48 both during a one to one chat or group chat.

User B is typing. As we observed the network activity of User A, we also observed the effects of typing by the other user (User B) of the IMO chat conversation. We found a slightly different behavior from the patterns observed when User A was typing. While User B was typing during the chat conversation, we observed a TCP pattern of the packets having a payload size of 144 bytes from the server. This pattern continued till User B finished the typing. In this case, a packet having a TCP payload size of 64 bytes was always sent from the target mobile (User A) in acknowledgement of each packet received from the IMO server as shown in Fig. 18.

Note that the packets with IDs [3594] and [3595] (Len = 144) from the server (actually User B at the other end) arrived so quickly that the User A didn't have the time to send the packet with a payload of 64 bytes in response to ID [3594] before the arrival of ID [3595]. So the User A sent packets with IDs [3597] and [3598] (Len = 64) in response to each packet later. It is a common scenario in TCP communication and it normally depends on the speed of the communication that how fast is the data being exchanged from one end to the other.

Call Initiated from User A. When the target mobile (User A) initiated a call, two consecutive packets having payload sizes of 272 bytes and 128 bytes respectively were sent to the server for establishing a call. In response, the IMO server acknowledged with a packet of 48 bytes in length. In less than a second, a series of packets were also sent by the server in which one of the packet size is greater than 1000 bytes and the others were always shorter in length as shown in Fig. 19.

In Fig. 19, packets with IDs 3915 is the response of packets having IDs [3911] and [3912] respectively received from the client. To detect the call initiation from User A, following pieces of evidence are clearly shown in Fig. 19:

- Any single packet from the server with a size of 48 bytes (after getting two consecutive packets having a payload sizes of 272 bytes and 128 bytes respectively).
- A packet of size greater than 1000 bytes must be received in less than a second from the server.

Call Initiated from User B. When the other user (User B) initiated a call, a packet having payload size of 112 bytes was received from the server and the target mobile (User A) acknowledged by sending a packet of 64 bytes in length. Then the server sent multiple packets in which one of the packet size was greater than 1000 bytes and the others were shorter in length. Afterwards, multiple packets were sent to the server for establishing a call in which two consecutive packet were of payload sizes 64 bytes and 176 bytes, respectively. This pattern of packet exchange between the target and the server occurred in less than 1.5 s, as shown in Fig. 20. Acknowledgement from the IMO server, in this case, was also of 48 bytes in length. Once the call was established, voice/video stream was carried over to UDP between the two users.

Following Fig. 20, packet ID [4206] shows 112 bytes received from server. Packet ID [4210] depicts the 64 bytes response of User A. 1348 bytes shown in packet ID [4214] is one of the larger packet sent by server and then packet IDs of [4224] and [4227] show consecutive packets of payload sizes of 64 bytes and 176 bytes which are sent from client to the IMO server.

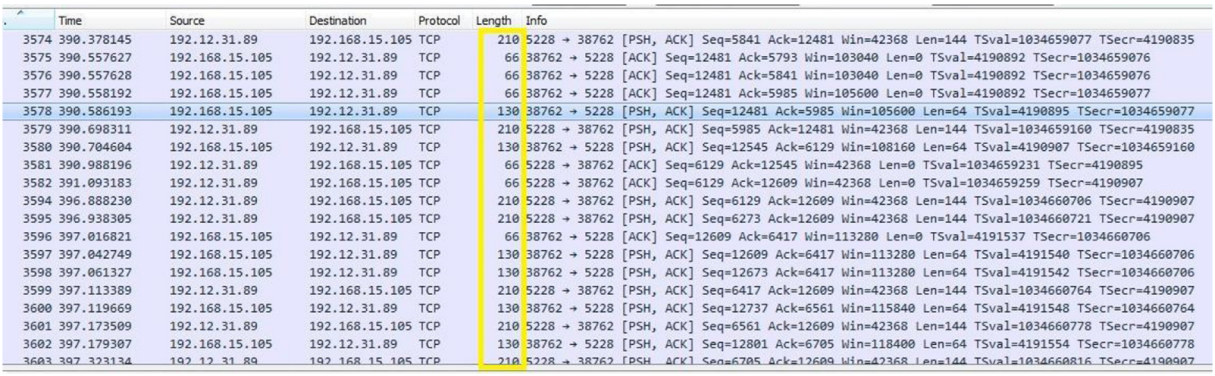
Correlating our findings of Figs. 19 and 20, the flow chart of call initiation by IMO client on Android which makes the identification of IMO call is depicted in Fig. 21.

Voice/video call termination. An important activity of IMO is detection of termination of voice or video call. The finish time of voice or video calls can easily be identified by looking at the UDP stream originating from the target IP after the detection of the call initiation events mentioned above. If the UDP stream flowing from the target IP stops, then the call will be considered as finished.

Table 3 summarizes our findings on traffic characteristic of IMO on Android.

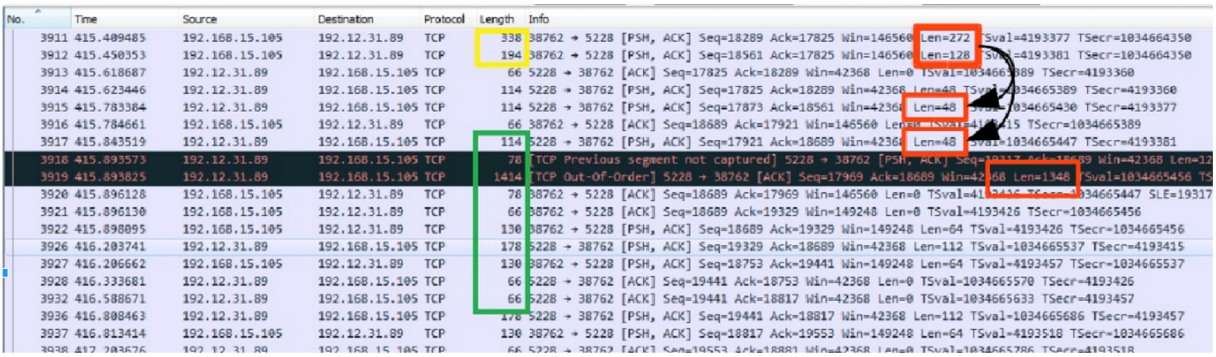
Traffic characteristics on iOS. Similarly, we performed behavior analysis on the IMO packets collected from iOS platform. The server ranges and TCP ports used in case of iOS are same as of Android but traffic flows between server and client possess different characteristics. Extensive investigation of payloads, packets lengths and frequency of traffic flows over the network were carried out against all the events described in case of Android and important conclusions were drawn to classify the IMO traffic on iOS platform.

User A is typing. When the target mobile (User A) was chatting with the other user (User B), pattern of TCP packets with fixed payload



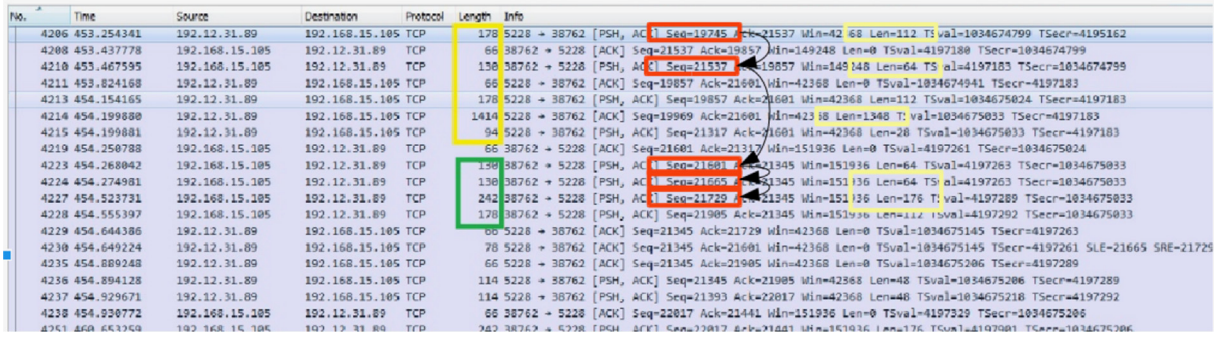
	Time	Source	Destination	Protocol	Length	Info
3574	390.378145	192.12.31.89	192.168.15.105	TCP	210	5228 → 38762 [PSH, ACK] Seq=5841 Ack=12481 Win=42368 Len=144 TSval=1034659077 TSecr=4190835
3575	390.557627	192.168.15.105	192.12.31.89	TCP	66	38762 → 5228 [ACK] Seq=12481 Ack=5793 Win=103040 Len=0 TSval=4190892 TSecr=1034659076
3576	390.557628	192.168.15.105	192.12.31.89	TCP	66	38762 → 5228 [ACK] Seq=12481 Ack=5841 Win=103040 Len=0 TSval=4190892 TSecr=1034659076
3577	390.558192	192.168.15.105	192.12.31.89	TCP	66	38762 → 5228 [ACK] Seq=12481 Ack=5985 Win=105600 Len=0 TSval=4190892 TSecr=1034659077
3578	390.586193	192.168.15.105	192.12.31.89	TCP	130	38762 → 5228 [PSH, ACK] Seq=12481 Ack=5985 Win=105600 Len=64 TSval=4190895 TSecr=1034659077
3579	390.698311	192.12.31.89	192.168.15.105	TCP	210	5228 → 38762 [PSH, ACK] Seq=5985 Ack=12481 Win=42368 Len=144 TSval=1034659160 TSecr=4190835
3580	390.704604	192.168.15.105	192.12.31.89	TCP	130	38762 → 5228 [PSH, ACK] Seq=12545 Ack=6129 Win=108160 Len=64 TSval=4190907 TSecr=1034659160
3581	390.988196	192.12.31.89	192.168.15.105	TCP	66	5228 → 38762 [ACK] Seq=6129 Ack=12545 Win=42368 Len=0 TSval=1034659231 TSecr=4190895
3582	391.093183	192.12.31.89	192.168.15.105	TCP	66	5228 → 38762 [ACK] Seq=6129 Ack=12609 Win=42368 Len=0 TSval=1034659259 TSecr=4190907
3594	396.888230	192.12.31.89	192.168.15.105	TCP	210	5228 → 38762 [PSH, ACK] Seq=6129 Ack=12609 Win=42368 Len=144 TSval=1034660706 TSecr=4190907
3595	396.938305	192.12.31.89	192.168.15.105	TCP	210	5228 → 38762 [PSH, ACK] Seq=6273 Ack=12609 Win=42368 Len=144 TSval=1034660721 TSecr=4190907
3596	397.016821	192.168.15.105	192.12.31.89	TCP	66	38762 → 5228 [ACK] Seq=12609 Ack=6417 Win=113280 Len=0 TSval=4191537 TSecr=1034660706
3597	397.042749	192.168.15.105	192.12.31.89	TCP	130	38762 → 5228 [PSH, ACK] Seq=12609 Ack=6417 Win=113280 Len=64 TSval=4191540 TSecr=1034660706
3598	397.061327	192.168.15.105	192.12.31.89	TCP	130	38762 → 5228 [PSH, ACK] Seq=12673 Ack=6417 Win=113280 Len=64 TSval=4191542 TSecr=1034660706
3599	397.113389	192.12.31.89	192.168.15.105	TCP	210	5228 → 38762 [PSH, ACK] Seq=6417 Ack=12609 Win=42368 Len=144 TSval=1034660764 TSecr=4190907
3600	397.119669	192.168.15.105	192.12.31.89	TCP	130	38762 → 5228 [PSH, ACK] Seq=12737 Ack=6561 Win=115840 Len=64 TSval=4191548 TSecr=1034660764
3601	397.173509	192.12.31.89	192.168.15.105	TCP	210	5228 → 38762 [PSH, ACK] Seq=6561 Ack=12609 Win=42368 Len=144 TSval=1034660778 TSecr=4190907
3602	397.179307	192.168.15.105	192.12.31.89	TCP	130	38762 → 5228 [PSH, ACK] Seq=12801 Ack=6705 Win=118400 Len=64 TSval=4191554 TSecr=1034660778
3603	397.323134	192.12.31.89	192.168.15.105	TCP	210	5228 → 38762 [PSH, ACK] Seq=6705 Ack=12609 Win=42368 Len=144 TSval=1034660816 TSecr=4190907

Fig. 18. User B typing – android.



No.	Time	Source	Destination	Protocol	Length	Info
3911	415.409485	192.168.15.105	192.12.31.89	TCP	338	38762 → 5228 [PSH, ACK] Seq=18289 Ack=17825 Win=146560 Len=272 TSval=4193377 TSecr=1034664350
3912	415.450353	192.168.15.105	192.12.31.89	TCP	194	38762 → 5228 [PSH, ACK] Seq=18561 Ack=17825 Win=146560 Len=128 TSval=4193381 TSecr=1034664350
3913	415.618687	192.12.31.89	192.168.15.105	TCP	66	5228 → 38762 [ACK] Seq=17825 Ack=18289 Win=42368 Len=0 TSval=1034665389 TSecr=4193360
3914	415.623446	192.12.31.89	192.168.15.105	TCP	114	5228 → 38762 [PSH, ACK] Seq=17825 Ack=18289 Win=42368 Len=48 TSval=1034665389 TSecr=4193360
3915	415.783384	192.12.31.89	192.168.15.105	TCP	114	5228 → 38762 [PSH, ACK] Seq=17873 Ack=18561 Win=42368 Len=48 TSval=1034665430 TSecr=4193377
3916	415.784661	192.168.15.105	192.12.31.89	TCP	66	38762 → 5228 [ACK] Seq=18689 Ack=17921 Win=146560 Len=0 TSval=4193415 TSecr=1034665389
3917	415.843519	192.12.31.89	192.168.15.105	TCP	114	5228 → 38762 [PSH, ACK] Seq=17921 Ack=18689 Win=42368 Len=48 TSval=1034665447 TSecr=4193381
3918	415.893573	192.12.31.89	192.168.15.105	TCP	78	[TCP Previous segment not captured] 5228 → 38762 [PSH, ACK] Seq=18113 Ack=18689 Win=42368 Len=12
3919	415.893825	192.12.31.89	192.168.15.105	TCP	1414	[TCP Out-Of-Order] 5228 → 38762 [ACK] Seq=17969 Ack=18689 Win=42368 Len=1348 TSval=1034665456 TS
3920	415.896128	192.168.15.105	192.12.31.89	TCP	78	38762 → 5228 [ACK] Seq=18689 Ack=17969 Win=146560 Len=0 TSval=4193426 TSecr=1034665447 SLE=19317
3921	415.896130	192.168.15.105	192.12.31.89	TCP	66	38762 → 5228 [ACK] Seq=18689 Ack=19329 Win=149248 Len=0 TSval=4193426 TSecr=1034665456
3922	415.898095	192.168.15.105	192.12.31.89	TCP	130	38762 → 5228 [PSH, ACK] Seq=18689 Ack=19329 Win=149248 Len=64 TSval=4193426 TSecr=1034665456
3926	416.203741	192.12.31.89	192.168.15.105	TCP	178	5228 → 38762 [PSH, ACK] Seq=19329 Ack=18689 Win=42368 Len=112 TSval=1034665537 TSecr=4193415
3927	416.206662	192.168.15.105	192.12.31.89	TCP	130	38762 → 5228 [PSH, ACK] Seq=18753 Ack=19441 Win=149248 Len=64 TSval=4193457 TSecr=1034665537
3928	416.333681	192.12.31.89	192.168.15.105	TCP	66	5228 → 38762 [ACK] Seq=19441 Ack=18753 Win=42368 Len=0 TSval=1034665570 TSecr=4193426
3932	416.588671	192.12.31.89	192.168.15.105	TCP	66	5228 → 38762 [ACK] Seq=19441 Ack=18817 Win=42368 Len=0 TSval=1034665633 TSecr=4193457
3936	416.808463	192.12.31.89	192.168.15.105	TCP	178	5228 → 38762 [PSH, ACK] Seq=19441 Ack=18817 Win=42368 Len=112 TSval=1034665686 TSecr=4193457
3937	416.813414	192.168.15.105	192.12.31.89	TCP	130	38762 → 5228 [PSH, ACK] Seq=18817 Ack=19553 Win=149248 Len=64 TSval=4193518 TSecr=1034665686
3938	417.203676	192.12.31.89	192.168.15.105	TCP	66	5228 → 38762 [ACK] Seq=19553 Ack=18881 Win=42368 Len=0 TSval=1034665786 TSecr=4193518

Fig. 19. Call initiated from target mobile – Android.



No.	Time	Source	Destination	Protocol	Length	Info
4206	453.254341	192.12.31.89	192.168.15.105	TCP	178	5228 → 38762 [PSH, ACK] Seq=19745 Ack=21537 Win=42368 Len=112 TSval=1034674799 TSecr=4195162
4208	453.437778	192.168.15.105	192.12.31.89	TCP	66	38762 → 5228 [ACK] Seq=21537 Ack=19857 Win=149248 Len=0 TSval=4197180 TSecr=1034674799
4210	453.467595	192.168.15.105	192.12.31.89	TCP	130	38762 → 5228 [PSH, ACK] Seq=21537 Ack=19857 Win=149248 Len=64 TSval=4197183 TSecr=1034674799
4211	453.824168	192.12.31.89	192.168.15.105	TCP	66	5228 → 38762 [ACK] Seq=19857 Ack=21601 Win=42368 Len=0 TSval=1034674941 TSecr=4197183
4213	454.154165	192.12.31.89	192.168.15.105	TCP	178	5228 → 38762 [PSH, ACK] Seq=19857 Ack=21601 Win=42368 Len=112 TSval=1034675024 TSecr=4197183
4214	454.199880	192.12.31.89	192.168.15.105	TCP	1414	5228 → 38762 [ACK] Seq=19969 Ack=21601 Win=42368 Len=1348 TSval=1034675033 TSecr=4197183
4215	454.199881	192.12.31.89	192.168.15.105	TCP	94	5228 → 38762 [PSH, ACK] Seq=21317 Ack=21601 Win=42368 Len=28 TSval=1034675033 TSecr=4197183
4219	454.250788	192.168.15.105	192.12.31.89	TCP	66	38762 → 5228 [ACK] Seq=21601 Ack=21317 Win=151936 Len=0 TSval=4197261 TSecr=1034675024
4223	454.268042	192.168.15.105	192.12.31.89	TCP	130	38762 → 5228 [PSH, ACK] Seq=21601 Ack=21345 Win=151936 Len=64 TSval=4197263 TSecr=1034675033
4224	454.274981	192.168.15.105	192.12.31.89	TCP	130	38762 → 5228 [PSH, ACK] Seq=21665 Ack=21345 Win=151936 Len=64 TSval=4197263 TSecr=1034675033
4227	454.523731	192.168.15.105	192.12.31.89	TCP	242	38762 → 5228 [PSH, ACK] Seq=21729 Ack=21345 Win=151936 Len=176 TSval=4197289 TSecr=1034675033
4228	454.555397	192.168.15.105	192.12.31.89	TCP	178	38762 → 5228 [PSH, ACK] Seq=21905 Ack=21345 Win=151936 Len=112 TSval=4197292 TSecr=1034675033
4229	454.644386	192.12.31.89	192.168.15.105	TCP	66	5228 → 38762 [ACK] Seq=21345 Ack=21729 Win=42368 Len=0 TSval=1034675145 TSecr=4197263
4230	454.649224	192.12.31.89	192.168.15.105	TCP	78	5228 → 38762 [ACK] Seq=21345 Ack=21601 Win=42368 Len=0 TSval=1034675145 TSecr=4197261 SLE=21665 SRE=21729
4235	454.889248	192.12.31.89	192.168.15.105	TCP	66	5228 → 38762 [ACK] Seq=21345 Ack=21905 Win=42368 Len=0 TSval=1034675206 TSecr=4197289
4236	454.894128	192.12.31.89	192.168.15.105	TCP	114	5228 → 38762 [PSH, ACK] Seq=21345 Ack=21905 Win=42368 Len=48 TSval=1034675206 TSecr=4197289
4237	454.929671	192.12.31.89	192.168.15.105	TCP	114	5228 → 38762 [PSH, ACK] Seq=21393 Ack=22017 Win=42368 Len=48 TSval=1034675218 TSecr=4197292
4238	454.930772	192.168.15.105	192.12.31.89	TCP	66	38762 → 5228 [ACK] Seq=22017 Ack=21441 Win=151936 Len=0 TSval=4197329 TSecr=1034675206
4251	460.653259	192.168.15.105	192.12.31.89	TCP	242	38762 → 5228 [PSH, ACK] Seq=22017 Ack=21441 Win=151936 Len=176 TSval=4197981 TSecr=1034675206

Fig. 20. Call initiated from other user – Android.

size was observed as shown in Fig. 22. At the beginning, packets with fixed TCP payload of 224 bytes were transferred from target mobile to IMO server. When the message size increased to a certain length, the payload size of TCP packets were observed to be increased by 16 bytes and this process of TCP pattern continued till the message was sent by the target. In the acknowledgement of each TCP packet sent by the target, a payload size of 96 bytes was received from the IMO server against each packet sent by the target. This unique pattern could be easily identified as the target mobile was having a chat with the other user.

In Fig. 7, packet ID [299] is the response of packet ID [291], packet ID [302] is the response of packet ID [292] and so on.

User B is typing. This behavior is slightly different from the pattern observed when the target mobile was typing. It started by receiving the packets having payload size of 240 bytes from the server and similarly the chunks of packets with the increased payload of 16 bytes kept coming till the other user finished typing. In this case, a packet having payload size of 112 bytes was sent from the target mobile in acknowledgement of each packet received from the IMO server as highlighted in Fig. 23.

Call initiated from User A. When the target mobile initiated a call, two consecutive packet having payload size of 304 bytes and 208 bytes respectively were sent to the server for establishing a call. In response of these messages, the server acknowledged with 96 bytes of packets against each packet sent by the target. In less than a second, a series of packets were also sent by the server in which one of the packet size was observed to be greater than 1000 bytes and the others were shorter in length (See Fig. 24). Once the call was established, voice/video stream was carried over UDP between the two users.

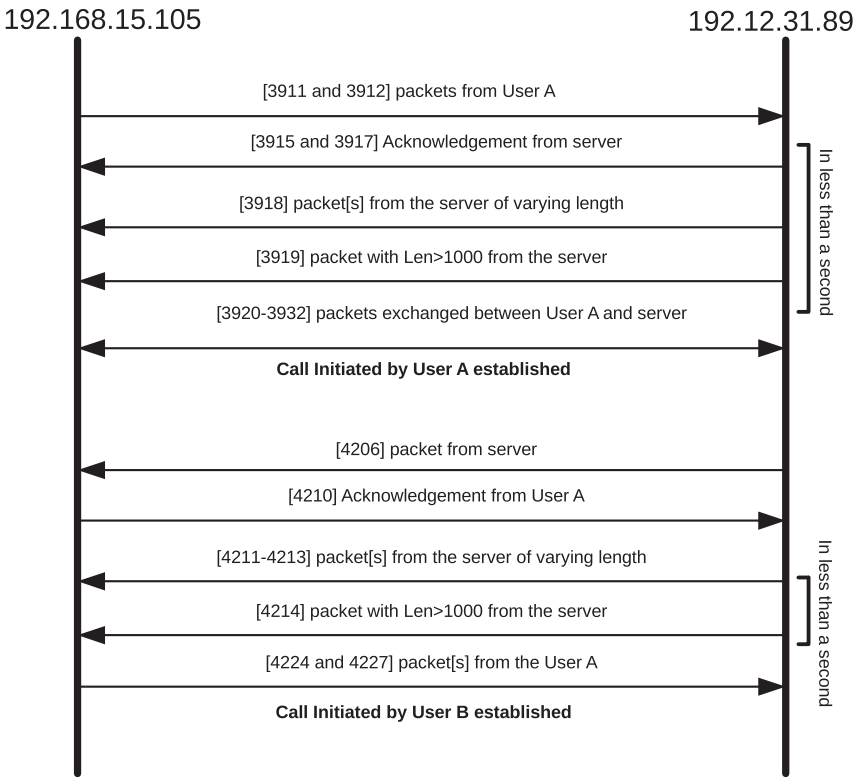


Fig. 21. Call initiation Flow – Android.

Table 3
Traffic characteristics of IMO on iOS.

Event	Description
Access to IMO messenger application	Same behavior as on Android
User A is typing	• Client **Beginning:** Packets with 224 bytes of payload from client to Server • Server 96 bytes of payload sent in acknowledgement of each packet sent from the target
User B is typing	• Server **Beginning:** Packets with 240 bytes of payload from Server to Target •Client 112 bytes of payload sent in acknowledgement of each packet sent from the server
Call initiated from User A	• Client Two consecutive packet having payload size of 304 bytes and 208 bytes respectively are sent to the server Or Two consecutive packet having payload size of 320 bytes and 192 bytes respectively are sent to the server Or Two consecutive packet having payload size of 320 bytes and 208 bytes respectively are sent to the server Or Two consecutive packet having payload size of 304 bytes and 272 bytes respectively are sent to the server • Server In response of these messages, the server acknowledges with 96 bytes of packets. In less than a second, a series of packets are also sent by the server in which one of the packet size is greater than 1000 bytes and the others are short in length
Call initiated from User B	• Server Packet having payload size of 224 bytes is received from the server and target mobile acknowledges by sending a packet of payload size 112 bytes • Client Target mobile acknowledges by sending a packet of payload size 112 bytes • Server In less than a second, a series of packets are also sent by the server in which one of the packet size is greater than 1000 bytes and the others are short in length
Voice/video call termination	UDP stream flowing from the target IP stops

Call initiated from User B. When the other user initiated a call, packet having payload size of 224 bytes was received from the server and target mobile acknowledged by sending a packet of payload size 112 bytes. Then the server sent multiple packets in which one of the packet size was greater than 1000 bytes and the others were shorter in length (See Fig. 25). Once the call was established, voice/video stream was carried over UDP between the two users.

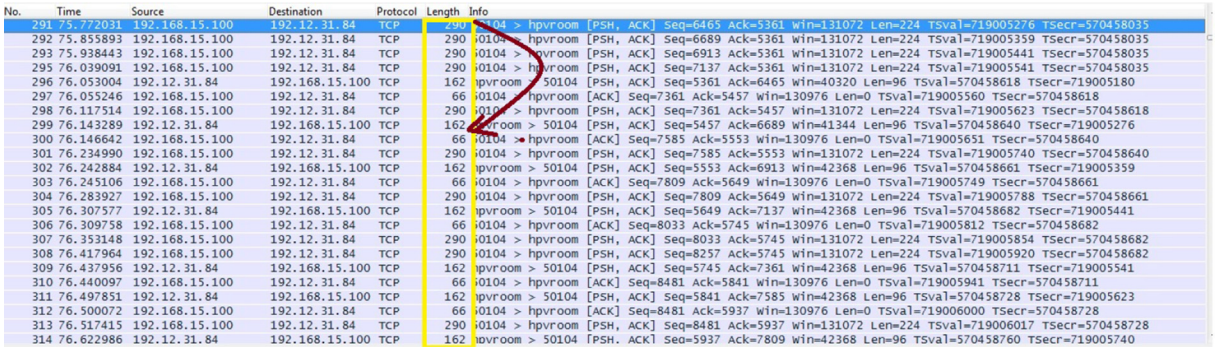


Fig. 22. User A Typing – iOS.

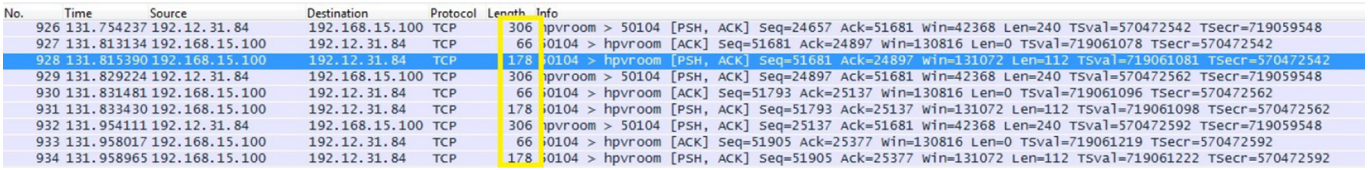


Fig. 23. User B Typing – iOS.

The flow chart of call initiation by IMO client on iOS which makes the identification of IMO call is depicted in Fig. 26.

Table 3 summarizes our findings of network traffic of IMO on iOS.

Crime scene reconstruction

In Section Device forensics of IMO, we showed that the user data is maintained in the device storage as plaintext. It is an interesting find, and can help an investigator to reconstruct a crime during an investigation involving mobile device forensics related to IMO artifacts. During network forensics of IMO, we have identified and outlined certain patterns in Section Network forensics of IMO that can help an investigator to attribute the communicating parties and to identify the type of communication taking place (chat or call). The following are the steps that an investigator should follow to gain insight into the case being investigated:

a) Gain an access to the suspect's network with IMO traffic.
b) Take the extensive traffic dumps for the duration of interest.
c) From traffic dumps, identify and filter IMO traffic based on results of server ranges and TCP ports as described in Section Server ranges and traffic behavior and Section TCP ports.
d) By using fixed chat and call patterns described in Section Network forensics of IMO, identify and filter out the complete traffic flows of users.
e) Archive filtered traffic for chats and voice/video calls separately.
f) After classification of events of IMO chats and voice/video calls, headers of payloads reveal IP addresses involved in these chat and voice/video calls.
g) From IP addresses, information on involved users can be obtained by taking assistance from Internet archives and concerned local Internet service providers.

With an assumption that investigator knows the IP address of User A, our study of crime scene reconstruction reveals that identification of User B from network traffic of IMO is only possible from

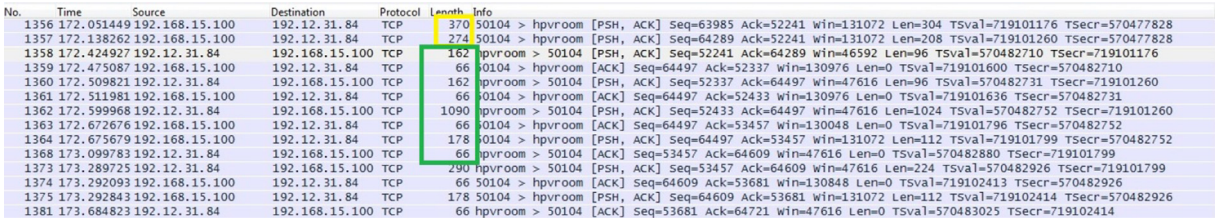


Fig. 24. Call Initiated by User A – iOS.

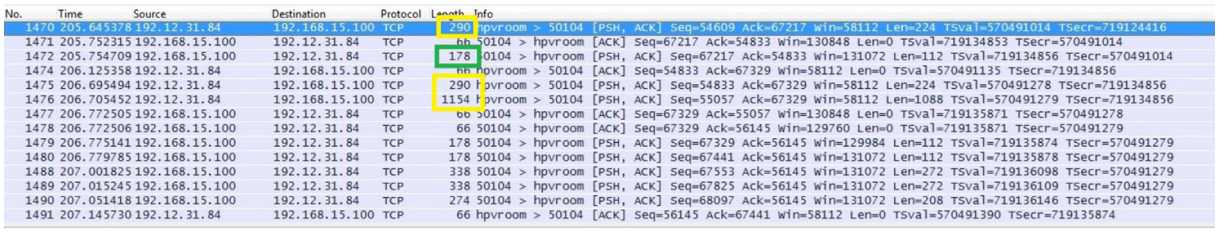


Fig. 25. Call Initiated by User B – iOS.

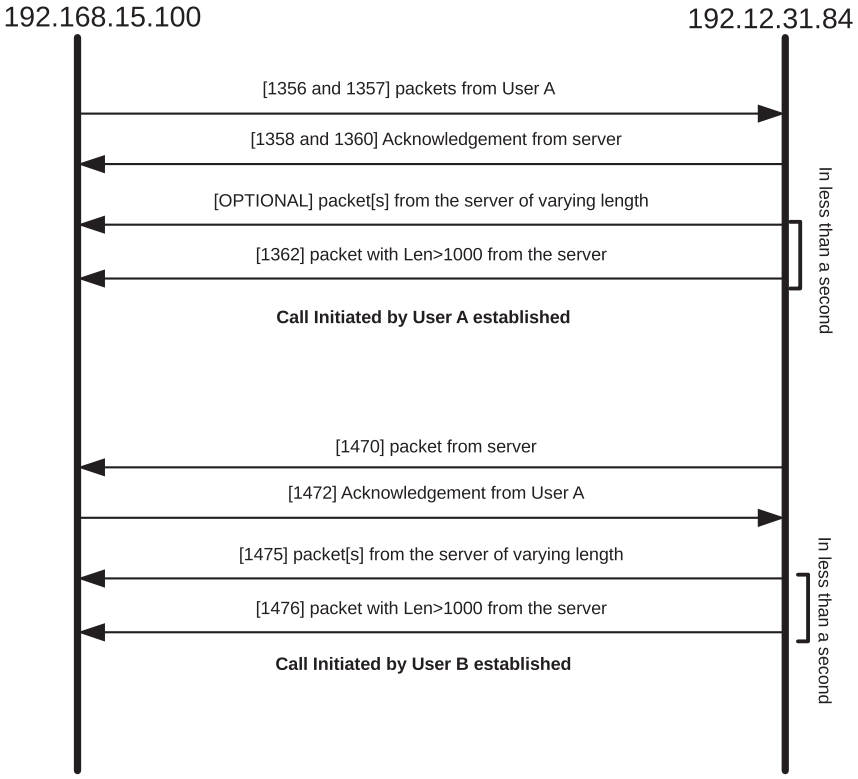


Fig. 26. Call initiation Flow – iOS.

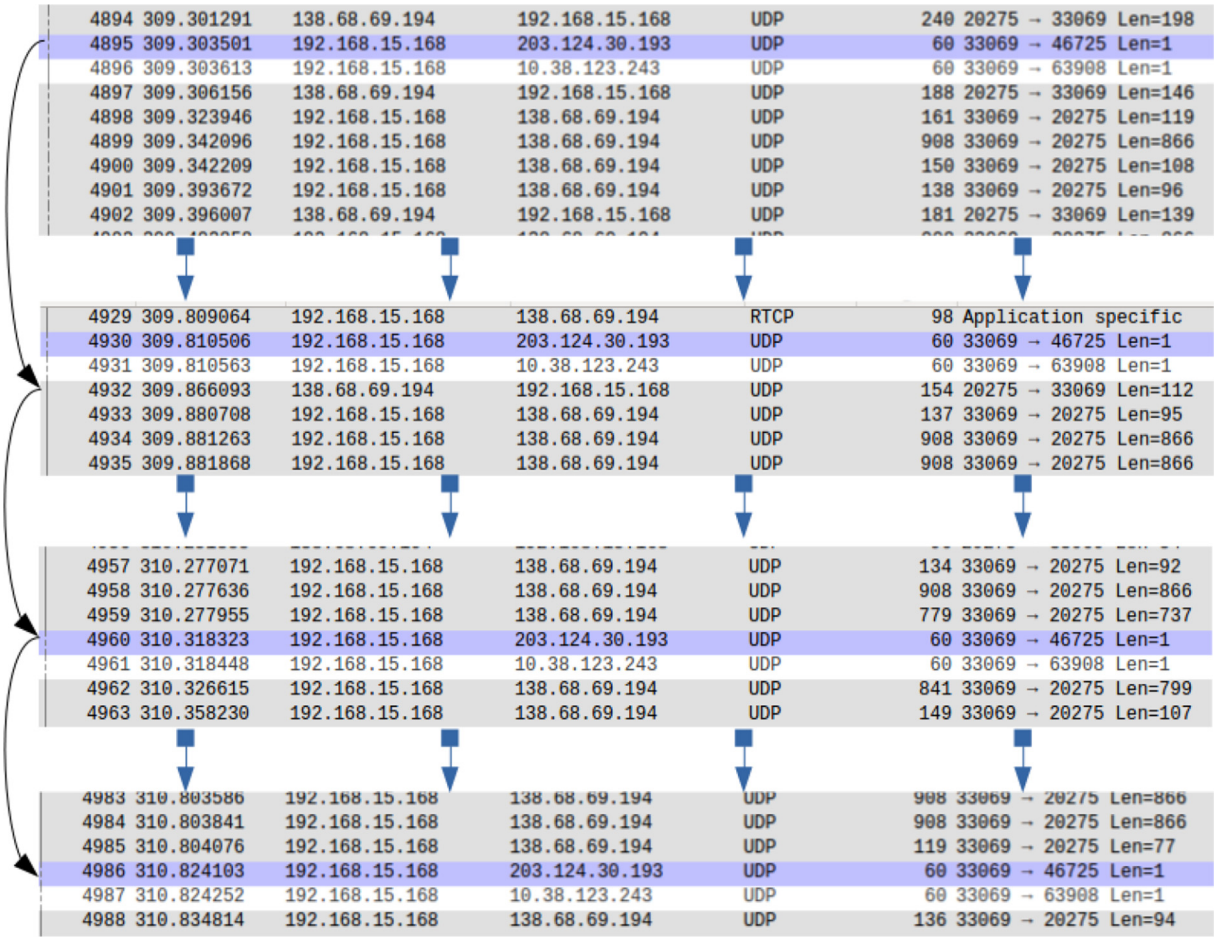


Fig. 27. IP address of User B.

UDP traffic flows of voice and video calls. After the successful initiation of voice/video calls through TCP flows, parallel sessions of UDP for establishing voice/video calls were observed to be attempted either through the IMO servers or directly to the User B. We observed that in most of the cases, voice/video calls were maintained through dedicated servers providing the voice/video services while attempt of direct connection with the User B was also made at intervals. We can safely assume that possibility of alternate connection to User B is also kept alive for uninterrupted voice/video service by IMO.

Fig. 27 depicts this scenario where voice call of User A (IP 192.168.15.168) was established to User B (IP 203.124.30.193) though IMO server providing the voice call services (138.68.69.194) while alternate connections were also attempted continuously with user B directly. This phenomena was observed at regular intervals during our study. It is important to mention that another IP address of 10.38.123.243 is also seen in Fig. 27, other than IP 203.124.30.193, which is not in a range of public IP addresses. We observed that the IP 10.38.123.243 is a private IP address while IP 203.124.30.193 is a public IP address of User B and same UDP ports are assigned to both of these IPs. Being in a private IP address range, we can simply discard private addresses while making the decisions during the course of our study. Filtering out the traffic of UDP can thus enable the investigator to identify both the calling parties of IMO voice/video calls.

Conclusion

In this paper, we conducted mobile device forensics and network forensics for IMO application. With no information on its communication protocol and security architecture, we explored possible vulnerabilities existing in the app design which can be exploited by an investigator to investigate a case involving IMO application. We were able to access the databases maintained by IMO app in mobile device both for Android and iOS. All useful information about the user was found in plain which could be easily extracted from a mobile device during the investigation of IMO. The critical user's information stored in unencrypted form includes phonebook entries, email Ids, call logs, chat logs and even the contents of the chats.

We also studied the network traffic of IMO extensively and drew significant results for correct traffic detection of IMO. A new idea of using firewall in such studies was demonstrated which supported our methodology to reveal obscured call connectivity scenarios of IMO. Our results on IMO traffic patterns can facilitate in correct detection of IMO over the network and further classification of events of IMO chats, voice and video calls. Different characteristics of IMO network traffic for Android and iOS were also highlighted in our work. The methodology we followed in this research is demonstrated by using IMO as a case study. It is important to note that this methodology is not specific for IMO only but can be generalized to other social media applications after a slight fine tuning in the ports and IP ranges of the associated servers.

Appendix A. Supplementary data

Supplementary data related to this article can be found at https://doi.org/10.1016/j.diin.2018.04.006.

References

Al Mutawa, N., Baggili, I., Marrington, A., 2012. Forensic analysis of social networking applications on mobile devices. Digit. Invest. 9, S24–S33.

Anglano, C., 2014. Forensic analysis of whatsapp messenger on android smartphones. Digit. Invest. 11 (3), 201–213.

Anglano, C., Canonico, M., Guazzone, M., 2016. Forensic analysis of the chatsecure instant messaging application on android smartphones. Digit. Invest. 19, 44–59.

Anglano, C., Canonico, M., Guazzone, M., 2017. Forensic analysis of telegram messenger on android smartphones. Digit. Invest. 23, 31–49.

AppAnnie, 2016. Snapchat and Imo Shake up the Most Popular Communication Apps. https://www.appannie.com/insights/app-annie-index-apps-may-2016-snapchat-imo/. (Accessed 16 March 2017).

Appelman, M., Bosma, J., Veerman, G., 2011. Viber Communication Security. System and Network of Engineering. University of Amsterdam, Netherlands.

Azfar, A., Choo, K.-K.R., Liu, L., 2016. An android communication app forensic taxonomy. J. Forensic Sci. 61 (5), 1337–1350.

Brunty, J., Miller, L., Helenek, K., 2014. Social Media Investigation for Law Enforcement. Routledge.

Chris, 2014. Helium. https://helium-sync-backup.en.softonic.com/android. (Accessed 3 June 2017).

Crunchbase, May 23, 2013. Izahrul — Crunchbase. https://www.crunchbase.com/organization/imo-im. (Accessed 16 March 2017).

Dupasquier, B., Burschka, S., McLaughlin, K., Sezer, S., 2010. Analysis of information leakage from encrypted skype conversations. Int. J. Inf. Secur. 9 (5), 313–325.

Eldon, E., 2007. Imo.Im, Offering Easy Video Chat through IM. http://venturebeat.com/2007/10/23/imo-im-offering-easy-video-chat-through-im/. (Accessed 16 March 2017).

GitHub, 2017. XMPP — Instant Messaging. https://xmpp.org/uses/instant-messaging.html. (Accessed 3 June 2017).

Gregorio, J., Gardel, A., Alarcos, B., 2017. Forensic analysis of telegram messenger for windows phone. Digit. Invest. 22, 88–106.

Huber, M., Mulazzani, M., Leithner, M., Schrittwieser, S., Wondracek, G., Weippl, E., 2011. Social snapshots: digital forensics for online social networks. In: Proceedings of the 27th Annual Computer Security Applications Conference. ACM, pp. 113–122.

Karpisek, F., Baggili, I., Breitinger, F., 2015. Whatsapp network forensics: decrypting and understanding the whatsapp call signaling messages. Digit. Invest. 15, 110–118.

Lillard, T.V., 2010. Digital Forensics for Network, Internet, and Cloud Computing: a Forensic Evidence Guide for Moving Targets and Data. Syngress Publishing.

Majeed, A., Zia, H., Imran, R., Saleem, S., 2015. Forensic analysis of three social media apps in windows 10. In: High-capacity Optical Networks and Enabling/Emerging Technologies (HONET), 2015 12th International Conference on. IEEE, pp. 1–5.

Marik, R., Bezpalec, P., Kucerak, J., Kencl, L., 2015. Revealing viber communication patterns to assess protocol vulnerability. In: 2015 International Conference on Computing and Network Communications (CoCoNet). IEEE, pp. 496–504.

Mehrotra, T., Mehtre, B., 2013. Forensic analysis of wickr application on android devices. In: Computational Intelligence and Computing Research (ICCIC), 2013 IEEE International Conference on. IEEE, pp. 1–6.

Molnár, S., Perényi, M., 2011. On the identification and analysis of skype traffic. Int. J. Commun. Syst. 24 (1), 94–117.

Norouzizadeh Dezfouli, F., Dehghantanha, A., Eterovic-Soric, B., Choo, K.-K.R., 2016. Investigating social networking applications on smartphones detecting facebook, twitter, linkedin and google+ artefacts on android and ios platforms. Aust. J. Forensic Sci. 48 (4), 469–488.

Ovens, K.M., Morison, G., 2016. Forensic analysis of kik messenger on ios devices. Digit. Invest. 17, 40–52.

ProVpnAccounts, May, 2016. List of Countries Which Block Website Content, VOIP and Skype. https://provpnaccounts.com/. (Accessed 17 March 2017).

Quora, May, 2016. In Which Countries Is Skype or WhatsApp Calling Blocked? https://www.quora.com/In-which-countries-is-Skype-or-WhatsApp-calling-blocked. (Accessed 17 March 2017).

Rubicon Communications, L, 2004-17. pfSense Appliances and Security Gateways. https://www.pfsense.org/products/. (Accessed 3 June 2017).

Seigfried-Spellar, K.C., Leshney, S.C., 2015. The intersection between social media, crime, and digital forensics:# whodunit? Digit. Forensics Threat. Best Pract. 59.

Taylor, M., Haggerty, J., Gresty, D., Almond, P., Berry, T., 2014. Forensic investigation of social networking applications. Netw. Secur. 2014 (11), 9–16.

Tso, Y.-C., Wang, S.-J., Huang, C.-T., Wang, W.-J., 2012. Iphone social networking for evidence investigations using itunes forensics. In: Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication. ACM, p. 62.

Walnycky, D., Baggili, I., Marrington, A., Moore, J., Breitinger, F., 2015. Network and device forensic analysis of android social-messaging applications. Digit. Invest. 14, S77–S84.

Wu, S., Zhang, Y., Wang, X., Xiong, X., Du, L., 2017. Forensic analysis of wechat on android smartphones. Digit. Invest. 21, 3–10.