Windows Instant Messaging App Forensics: Facebook and Skype as case studies

Yang, TY, Dehghantanha, A, Choo, KR and Muda, Z

http://dx.doi.org/10.1371/journal.pone.0150300

<table>
<thead>
<tr>
<th>Title</th>
<th>Windows Instant Messaging App Forensics: Facebook and Skype as case studies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authors</td>
<td>Yang, TY, Dehghantanha, A, Choo, KR and Muda, Z</td>
</tr>
<tr>
<td>Type</td>
<td>Article</td>
</tr>
<tr>
<td>URL</td>
<td>This version is available at: <a href="http://usir.salford.ac.uk/37999/">http://usir.salford.ac.uk/37999/</a></td>
</tr>
<tr>
<td>Published Date</td>
<td>2016</td>
</tr>
</tbody>
</table>

USIR is a digital collection of the research output of the University of Salford. Where copyright permits, full text material held in the repository is made freely available online and can be read, downloaded and copied for non-commercial private study or research purposes. Please check the manuscript for any further copyright restrictions.

For more information, including our policy and submission procedure, please contact the Repository Team at: usir@salford.ac.uk.
RESEARCH ARTICLE

Windows Instant Messaging App Forensics: Facebook and Skype as Case Studies

Teing Yee Yang¹, Ali Dehghantanha², Kim-Kwang Raymond Choo³*, Zaiton Muda¹

1 Department of Computer Science, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, UPM Serdang, Selangor, Malaysia, ² The School of Computing, Science & Engineering, Newton Building, University of Salford, Salford, Greater Manchester, United Kingdom, ³ Information Assurance Research Group, University of South Australia, Adelaide, South Australia, Australia

* raymond.choo@fulbrightmail.org

Abstract

Instant messaging (IM) has changed the way people communicate with each other. However, the interactive and instant nature of these applications (apps) made them an attractive choice for malicious cyber activities such as phishing. The forensic examination of IM apps for modern Windows 8.1 (or later) has been largely unexplored, as the platform is relatively new. In this paper, we seek to determine the data remnants from the use of two popular Windows Store application software for instant messaging, namely Facebook and Skype on a Windows 8.1 client machine. This research contributes to an in-depth understanding of the types of terrestrial artefacts that are likely to remain after the use of instant messaging services and application software on a contemporary Windows operating system. Potential artefacts detected during the research include data relating to the installation or uninstallation of the instant messaging application software, log-in and log-off information, contact lists, conversations, and transferred files.

1. Introduction

Instant messaging (IM) is popular with both traditional computing device users (i.e., personal computers and laptops) and mobile device users by allowing them to exchange information with peers in real time using text messaging, voice messaging, and file sharing. According to the report of Radicati Group [1], the number of worldwide IM accounts (with the exception of mobile messaging) in 2015 amounted to over 3.2 billion which is expected to rise above 3.8 billion by the end of 2019.

Similar to other popular consumer technologies, IM services have also been exploited to commit frauds and scams [2–4], disseminate malware [5], groom children online with the purpose of sexual exploitation [6–9] etc. The chat logs can provide a great deal of information of evidential value to investigators [10, 11], which may often comprise a suspect’s physical location, true identity, transactional information, incriminating conversations, and other person information i.e., email address and bank account number [12].

Due to the increased user privacy requirements [13] and demands for data redundancy, it is increasingly challenging to collect evidential data from the IM service provider (ISP). The data
are often protected by proprietary protocols, encryption, etc., making forensic practitioners virtually impossible to collect meaningful information from external network [14]. Moreover, collecting data from a multi-tenancy environment may breach the data privacy policies of the ISPs [15]. Even if the artefacts could be identified, the challenges are compounded by cross-jurisdictional investigations that may prohibit cross-border transfer of information [16–18]. In the worst-case scenario, the ISPs may not even log the incriminating conversations to reduce traffic to the messaging servers [19].

Depending on the IM application in use, the client device can often provide potential for alternative methods for recovery of the IM artefacts [20–22]. In addition to addressing the possible issues in relation to evidence acquisition from the ISPs, the terrestrial artefacts can be useful in establishing whether a suspect has a direct connection to a crime, as the suspect may claim he/she is a victim of identity theft otherwise. While a practitioner should be cognisant of techniques of digital forensics, it is just as important to maintain an up-to-date understanding of the potential artefacts that are recoverable from different types of IM products. Hence, in this paper, we seek to identify potential terrestrial artefacts that may remain after the use of the popular Facebook and Skype Windows Store application software (henceforth the Store app) on a Windows 8.1 client machine. Similar to the approaches of Quick and Choo [23–25], we attempt to answer the following questions in this research:

1. What data remains on a Windows 8.1 device and their locations on a hard drive after a user has used Facebook app version 1.4.0.9 and Skype app version 3.1.0.1007.

2. What data remains in Random Access Memory (RAM) after a user has used the above IM services or apps on a Windows 8.1 device?

3. What data can be seen in network traffic?

Findings from this research will contribute to the forensic community’s understanding of the types of terrestrial artefacts that are likely to remain after the use of IM services and apps on devices running the newer Windows operating system.

The structure of this paper is as follows. Section 2 discusses the background and related work. Section 3 outlines the research methodology and experiment environment and setup. In Sections 4 to 6, we present and discuss the findings from the IM apps. We then conclude the paper and outline potential future research areas in the last section.

2. Literature Review

A Windows Store app (formerly known as Metro app) mimics the touch-screen-friendly mobile apps, while retaining the traditional mouse and keyboard inputs [26]. The installation is handled exclusively by the Windows Store, which bypasses the execution of executable files [27]. The Store apps are licensed to Microsoft account, giving the users the right to install a same app on up to eighty-one different Windows 8 (or newer) desktop clients under the same login [28]. The concept also enables the users to roam the app credentials (stored within the Credential Locker) between the corresponding devices [29].

The Store apps are predominantly built on Windows Runtime. In addition to offering the developers a multi-language programming environment, the architecture isolates the apps from the file system for security and stability [26]. The app itself is a package (.APPX file) that incorporates the app’s code, resources, libraries, and a manifest up to a combined limit of 8GB [26]. Each Store app is represented by a package ID, which is often denoted by the package name followed by its build version, the target platform, and the alphanumeric publisher identification (ID). The installation and application folders can be generally located in %Program
The application data, correspond to the app states [26], are stored in three (3) categories: local, roaming, and temp states; each of which creates a subfolder in the application folder. The 'LocalState' folder holds device-specific data typically loaded to support the app functionality, such as temporary files and caches, recently viewed items, and other behavioural settings. The 'RoamingState' folder stores data shared between the same app running on multiple Windows devices under the same login. The data may include account configurations, favourites, game scores and progress, important URIs etc. Meanwhile, the 'TempState' folder houses data temporarily suspended or terminated from the memory for restoration purposes, such as page navigation history, unsaved form data etc. The application data persist throughout the lifetime of a Store app, with the exception of the temp data which may be subject to disk clean up [26].

The application cache/data can be stored using caching mechanisms like HTML5 local storage and IndexedDB (for Store apps written in HTML and JavaScript) as well as other third-party database options like SQLite [32]. In the absence of encryption mechanism, the data can aid in reconstruction of user events such as cloud storage [28], emails [30], web browsing history [33], conversations [34], and other user-specific events [35], depending on the Store app in use.

Instant messaging has been the subject of numerous digital forensic studies since the mid 2000’s. In a series of early works, Dickson identified that artefacts of the client-based American Online Messenger version 5.5 (AIM) [16], MSN Messenger version 7.5 [36], Yahoo Messenger version 7.0 [37], and Trillian version 3.1 [38] could be recovered from the registry, user settings, and other application-specific files on the hard drive of a Windows XP machine. By applying keyword search, the author was able to recover portion of the conversation history from unstructured datasets such as memory dumps, slack space, free space, and swap files in plain text, even with the absence of chat logging. The findings were echoed by several others studies with respect to Digsby [39–41], Windows Live Messenger 8.0 [42], and Pidgin 2.0 [43]. However, Levendoski et al. [44] concluded that artefacts of the Yahoo Messenger client produced a different directory structure on Windows Vista/7. Kiley et al. [19] investigated web-based IM apps (i.e., AIM Express, Google Talk, Meebo, and E-Buddy) and found that artefacts of the contact lists, conversations, and approximate time of the last conversation could only be recovered from memory dump and hard disk’s free space, although reference to the URLs, last access times, and view count information could be recovered from the web browsing history.

Wong et al. [45] and Al Mutawa et al. [46] demonstrated that artefacts of the Facebook web-application could be recovered from memory dumps and web browsing cache in JavaScript Object Notation (JSON) and Hypertext Markup Language (HTML) formats. Al Mutawa et al. [46] also described a methodology for investigating the Arabic string artefacts on a computer device. In another study, Al Mutawa et al. [47] investigated artefacts of the Facebook and several other IM applications on iPhone 4, Blackberry Torch 9800, and Samsung GT-i9000 Galaxy S. The authors were able to extract records of the contact list and conversation from the logical images, with the exception of the BlackBerry devices.

Said et al. [48] investigated Facebook and other IM applications for iPhone 3G and 3GS, Blackberry Bold 7000 and 900, Samsung Omnia II i8000, Nokia E71, and Ericsson G900. Of all the mobile devices investigated, it was determined that only BlackBerry Bold 9700 and iPhone 3G/3GS provided evidence of Facebooking unencrypted. The study also revealed that artefacts of the Facebook applications were unique to the mobile devices investigated (i.e., iPhone 3GS and iphone 3G had the same version of Facebook v3.4.2 but maintained different files in the application folders). Walnycky et al. [49] added that artefacts of the Facebook Messenger could vary depending on user settings, OS version, and manufacturer. Levinson et al. [50]
demonstrated that records of the recent Facebook chats stored in the property list of the Face-
book Messenger for iOS can assist forensic practitioners with timeline analysis.  
Exchanging iTunes backups rather than disk images, Norouzizadeh et al. [10] and Tso et al. [51] concluded that it is possible to extract users’ personal data, messages, contact lists and  
posts Facebook app from the iTunes backup of iPhone 4 and iPhone 5s, respectively. Chu et al. [52] focused on live data acquisition from the desktop personal computer (PC) and was able to  
identify distinct strings that will assist forensic practitioners with reconstruction of the previous Facebook sessions. Wongyai and Charoenwatana [53] determined that objects recovered from  
a network analysis of Facebook homepage can be broadly categorised into 24 types based on properties such as file type, naming pattern, IP address, and location or section on the page.  
Sgaras et al. [54] analysed Skype and several other VoIP applications for iOS and Android  
platforms. Although footprints of the installations, user profiles, conversations, contact lists, and network traffic could be located for all the VoIP applications investigated, it was concluded that the Android apps store far less artefacts than of the iOS apps. Simon and Slay [55] found that remnants of Skype communication, communication history, contacts, passwords, and  
encryption keys could be recovered from physical memory dump. However, Teng and Lin [56]  
demonstrated that using SQLite editor tools, one could easily modify Skype log files. Unsur-
prisingly, other studies have suggested that the network traffic behaviour varies among differ-
ent versions [57, 58].  
In the only article on Windows Store apps for instant messaging (at the time of this  
research), Lee and Chung [34] studied the third party Viber and Line apps and identified that the package identifications (IDs) could be discerned from ‘2414F_C7A.ViberFreePhoneCall-
sText_p61zvh252yqr’ and ‘NA_VER.LINEwin8_8ptj331gd3tyt’ respectively. By analysing the  
app caches, the authors managed to locate records of account logins, contacts, chats, trans-
ferred file unencrypted. However, the study is only limited to dead analysis of the hard disk.  
Hence, there is a need to develop a further understanding of the implications of the Windows  
Store apps for IM forensics—a gap that this paper aims to contribute to.

3. Research Methodology

The examination procedure in this research is adapted from the four-stage digital forensic  
framework of McKemmish [59], namely: identification of digital evidence, preservation of digi-
tal evidence, analysis, and presentation. The purpose is to enable acquisition of realistic data  
similar to that found in real world investigations. This paper mainly focuses on the analysis  
stage, although we also briefly discuss the evidence source identification, preservation, and pre-
sentation to demonstrate how the framework could be applied in practice.

The first step of the experiment involved the creation of eight (8) fictional accounts to play  
the role of suspects and victims in this research—see Table 1. The IM accounts were assigned  
with a unique ‘display icon’ and username which was not used within the respective IM apps  
and Windows operating system. This eases identification of the user roles. Next was to create  
the test environments for the suspects and the victims, which consisted two (2) control base  
VMware Workstations (VMs) version 9.0.0 build 812388 running Windows 8.1 Professional  
(Service Pack 1, 64 bit, build 9600). As explained by Quick and Choo [23–25], using physical  
hardware to undertake setup, erasing, copying, and re-installing would have been an onerous  
exter. Moreover, a virtual machine allows room for error by enabling the test environment  
to be reverted to a restore point should the results are unfavourable. The workstations were  
configured with the minimal space (2GB of physical memory and 20GB hard drive space) in  
order to reduce the time required to analyse the considerable amounts of snapshots in the latter  
stage.
In the third step, we conducted a predefined set of activities to simulate various real world scenarios of using the apps on each workstation/test environment. The base assumptions are that the practitioner encounters a live system running Microsoft Windows 8.1 in a typical home environment. Similar to the approaches of Quick and Choo [23–25], the 3111th email message of the University of California (UC) Berkeley Enron email dataset (downloaded from http://bailando.sims.berkeley.edu/enron_email.html on 24th September 2014) was used to create the sample files and saved as SuspectToVictim.rtf, SuspectToVictim.txt, SuspectToVictim.docx, SuspectToVictim.zip, SuspectToVictim.jpg (printscreen), VictimToSuspect.rtf, VictimToSuspect.txt, VictimToSuspect.docx, VictimToSuspect.jpg (printscreen), and VictimToSuspect.zip to simulate the transferring and receiving of files of different formats using the IM apps. As the filenames suggest, the ‘SuspectToVictim’ (and ‘VictimToSuspect’) files were placed on the suspect’s workstation (and victims’ workstations respectively) and subsequently transferred to the victims’ workstations (and suspect’s workstation respectively).

The experiments were predominantly undertaken in NATed (where NAT stands for Network Address Translation) network environment and without firewall outbound restriction to represent a typical IM situation. Wireshark was deployed on the host machine to capture the network traffic from the suspect’s workstation for each scenario. After each experiment was carried out, we saved a copy of the network capture file in.PCAP format, and acquired a bit-stream (dd) image of the virtual memory (.VMEM) file prior to shutdown. We then took a snapshot of each workstation after being shutdown and made a forensic copy of the virtual disk (.VMDK) file in Encase Evidence (E01) format. This resulted in the creation of fifteen (15) snapshots (each for each environment) as highlighted in Table 2, and Figs 1 and 2. The decision to instantiate the physical memory dumps and hard disks with the virtual disk and memory files was to prevent the datasets from being modified with the use of memory/image acquisition tools [23, 25].

The final step of this research was to analyse the datasets using a range of forensically recognised tools (as highlighted in Table 3) and present the findings. Both indexed and non-indexed as well as Unicode and non-Unicode string searches were included as part of the evidence searches. The experiments were repeated at least thrice (at different dates) to ensure consistency of findings.

### Table 1. Account details for IM experiments.

<table>
<thead>
<tr>
<th>IM Experiment</th>
<th>Username</th>
<th>Email</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook Messenger Forensics</td>
<td>John Adkins</td>
<td><a href="mailto:johnadkins109@gmail.com">johnadkins109@gmail.com</a></td>
<td>Suspect</td>
</tr>
<tr>
<td></td>
<td>Jack Jeffry</td>
<td><a href="mailto:fbcchelper@gmail.com">fbcchelper@gmail.com</a></td>
<td>Suspect 2</td>
</tr>
<tr>
<td></td>
<td>Adam Jacobs</td>
<td><a href="mailto:adamjacobs717@gmail.com">adamjacobs717@gmail.com</a></td>
<td>Victim 1</td>
</tr>
<tr>
<td></td>
<td>Samuel Traviss</td>
<td><a href="mailto:samueltraviss@gmail.com">samueltraviss@gmail.com</a></td>
<td>Victim 2</td>
</tr>
<tr>
<td></td>
<td>Kelvin Sky</td>
<td><a href="mailto:fbctester@gmail.com">fbctester@gmail.com</a></td>
<td>Victim 3</td>
</tr>
<tr>
<td>Skype Forensics</td>
<td>Adam Thomson/</td>
<td><a href="mailto:adamthomson1984@gmail.com">adamthomson1984@gmail.com</a></td>
<td>Suspect</td>
</tr>
<tr>
<td></td>
<td>Harold Cornwall/</td>
<td><a href="mailto:haroldcwall@gmail.com">haroldcwall@gmail.com</a></td>
<td>Victim 1</td>
</tr>
<tr>
<td></td>
<td>Alicia Richardson/</td>
<td><a href="mailto:aliciarich19@gmail.com">aliciarich19@gmail.com</a></td>
<td>Victim 2</td>
</tr>
</tbody>
</table>

4. Analysis of the Facebook App

Facebook (Messenger) is an IM service offered by Facebook—one of the most popular social network platforms with more than one billion daily active users on average [60]. The Store app was officially released on 17th October 2013 in conjunction with the launch of Windows 8.1 [61]. It allows users to view status updates, news feeds, send and receive text and voice, as well
as features such as file transfer and image sharing. In this section, we present artefacts of installation, uninstallation, logins, contact lists, conversations, transferred files, and notifications of the Facebook app (version 1.4.0.9) on Windows 8.1.

Table 2. Details of VM snapshots created for this research.

<table>
<thead>
<tr>
<th>IM forensics</th>
<th>Snapshot</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 Base-Snapshot</td>
<td>A control base snapshot was made to create the control media to determine changes from each IM scenario.</td>
<td></td>
</tr>
<tr>
<td>Facebook forensics</td>
<td>F1.1 Install-Snapshot</td>
<td>Using a duplicate copy of the control base snapshot (1.0), we accessed the Windows Store to download and subsequently install the Facebook app version 1.4.0.9.</td>
</tr>
<tr>
<td></td>
<td>F1.1.1 Login-Snapshot</td>
<td>A snapshot was made of the install snapshot (F1.1) to examine the artefacts from the Facebook login.</td>
</tr>
<tr>
<td></td>
<td>F1.1.2 Friend-Snapshot</td>
<td>A second snapshot was created of the install snapshot (F1.1) to examine the process of searching and adding friend using the Facebook app.</td>
</tr>
<tr>
<td></td>
<td>F1.1.3 Chat-Snapshot</td>
<td>Another snapshot was made of the install snapshot (F1.1) to undertake scripted conversations and file transfers using the Facebook app. The conversations were limited to two participants.</td>
</tr>
<tr>
<td></td>
<td>F1.1.4 Group Chat-Snapshot</td>
<td>A final snapshot was made of the install snapshot (F1.1) to examine the artefacts left by the group chat feature of the Facebook app. The suspect's account was used to add all the victims into a group chat namely ‘DeviGroup’. A mock conversation was subsequently taken between the suspect and the victims.</td>
</tr>
<tr>
<td>Skype forensics</td>
<td>S1.1 Install-Snapshot</td>
<td>Using a duplicate copy of the control base snapshot (1.0), we updated the Skype app to version 3.1.0.1007 (the latest version at the time of this research).</td>
</tr>
<tr>
<td></td>
<td>S1.1.1 Login-Snapshot</td>
<td>A snapshot was made of the install snapshot (S1.1) to examine the login artefacts of the Skype app.</td>
</tr>
<tr>
<td></td>
<td>S1.1.2 Contact-Snapshot</td>
<td>A second snapshot was made of the install snapshot (S1.1) to examine the process of adding contact using the Skype app. The contacts were subsequently synced to the Windows Live (login) account to identify additional artefacts in relation to the contact syncing.</td>
</tr>
<tr>
<td></td>
<td>S1.1.3 IM-Snapshot</td>
<td>A third snapshot was made of the install snapshot (S1.1) to undertake scripted IM conversations and file transfers using the Skype app. The conversations were limited to two participants.</td>
</tr>
<tr>
<td></td>
<td>S1.1.4 Group Chat-Snapshot</td>
<td>Another snapshot was created of the install snapshot (S1.1) to examine the group chat artefacts of the Skype app. The suspect's account was used to add all the victims into a group chat namely ‘DeviSkypeGroup’. A mock conversation was subsequently taken between the suspect and the victims.</td>
</tr>
<tr>
<td></td>
<td>S1.1.5 Voice and Video Call-Snapshot</td>
<td>Additional copy of the install snapshot (S1.1) was made to examine the process of voice and video calling using the Skype app. We first made a Skype to Skype voice call from the suspect to victim, followed by a video call during the voice call.</td>
</tr>
<tr>
<td></td>
<td>S1.1.6 Video Message-Snapshot</td>
<td>A final copy of the install snapshot (S1.1) was made to investigate the process of creating video message using the Skype app. A video message was made and subsequently sent from the suspect's VM to the victim's VM.</td>
</tr>
</tbody>
</table>

doi:10.1371/journal.pone.0150300.t002

Fig 1. VM snapshots created for Facebooking experiments.

doi:10.1371/journal.pone.0150300.g001
4.1 Installation of the Facebook App

Examinations of the directory listings observed that the package ID (for the Facebook app) can be differentiated from 'Facebook.Facebook_1.4.0.9_x64__8xx8rvfyw5nnt'. A closer examination of the registry entries created during the installation observed that the installation time could be identified from the 'InstallTime' entry within the HKEY_USERS\<SID>\Software \Classes\Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\Repository \Families\Facebook\Facebook_8xx8rvfyw5nnt\Facebook\Facebook_1.4.0.9_x64_8xx8rvfyw5nnt branch in 64-bit FILETIME Hex value in Big Endian format.

A search for the package ID 'Facebook.Facebook_1.4.0.9_x64__8xx8rvfyw5nnt' in the Windows Store logs (resided at %AppData%\Local\Temp\winstore.log and %AppData%\Local \Packages\winstore_cw5n1h2xyewy\AC\Temp\winstore.log) located supporting timestamp information such as the dates when the app was first launched and updated. Moreover, analysis of the prefetch files revealed the last run time and number of times the app has been loaded in 'FACEBOOK.EXE.pf'. As for event logs, there was additional timestamp information which indicated the accessed times in 'Application.evtx', 'Microsoft-WS-Licensing%4Admin.evtx', 'Microsoft-Windows-AppModel-Runtime%4Admin.evtx', 'Microsoft-Windows-AppDeploymentServer%4Operational.evtx', 'Microsoft-Windows-Audio%4PlaybackManager.evtx', 'Microsoft-Windows-CoresApplication%4Operational.evtx', 'Microsoft-Windows-PushNotification-Platform%4Operational.evtx', 'Microsoft-Windows-Resource-Exhaustion-Resolver% 4Operational.evtx', 'Microsoft-Windows-SettingSync%4Debug.evtx', 'Microsoft-Windows-Shell-Core%4Operational.evtx', 'Microsoft-Windows-TWinUI%4Operational.evtx', 'Microsoft-Windows-Windows Firewall With Advanced Security%4Firewall.evtx', and 'System.evtx'.

Examinations of the running processes using the 'pslist' function of Volatility determined that the process name could be discerned from 'Facebook.exe'. Fig 3 illustrates that the 'pslist' output also included the process identifier (PID), parent process identifiers (PPID), and the process initiation and termination time. The PID could prove useful for correlating data associated with the app during further analysis of the RAM (i.e., contextualising a string using the 'Yarascan' function of Volatility).
In our experiments, it was observed that Facebook maintains a wealth of cache data for the Store app in a number of SQLite databases located in %AppData%\Local\Packages\Facebook.Facebook_1.4.0.9_x64__8xx8rwy5mnt\LocalState\<User specific Facebook ID>\DB, such as Analytics.sqlite, FriendRequests.sqlite, Friends.sqlite, Messages.sqlite, Notifications.sqlite, and Stories.sqlite. However, it is noteworthy that these databases will only appear when the user is logged in from the app. The database of interest with the logins is Analytics.sqlite, which contains records of the login time in Unix epoch format. The records can be discerned from the 'name' and 'module' table columns which reference 'login' and 'login_events' in the 'analytics_s_logs' table, respectively—see Fig 4. Within %AppData%\Local\Packages\Facebook.Facebook_8xx8rwy5mnt\AC\InetCache\<Cache ID>\ and %AppData%\Local\Packages\Facebook.Facebook_8xx8rwy5mnt\AC\local_cache\ there were copies of profile and cover pictures of the user and the contacts, as well as other pictures which appeared on the Facebook timelines. The pictures may provide invaluable leads that lay the groundwork for follow-up via traditional investigative techniques.
A search for the login password produced no matches in the forensic image and memory dump. An examination of the network traffic revealed that the host first established a session with Symantec Certification Authority (i.e., IP address 23.58.43.27) for certificate authentication. Afterwards, the host accessed the nearest Akamai content delivery servers (i.e., IP addresses 23.62.109.* and 31.13.* in our research) and Facebook servers from different countries (i.e., IP addresses 115.164.13.*) on port 443 (hence HTTPS), which we theorised to retrieve the profile and timeline information. Although the network traffic was encrypted and the login credentials were not recovered, we were able to correlate the IP addresses with the timestamp information to determine when the app was started up and the duration of Facebook use in our research.

4.3 Friend Lists

Contact (or ‘friend’ in the context of Facebook) lists can be a useful reference point for a suspect’s social network. A search for the suspect’s profile name in the directory listing determined that artefacts of the contact lists can only be located in the Friends.sqlite database. The table of particular interest is the ‘friends’ table, which holds a list of user identifications (UIDs), full names, first names, middle names, last names, email addresses, phone numbers, profile links, communication rank (frequency of communication), and birth dates associated with the friends added by the user as shown in Fig 5.

Moreover, the ‘profiles’ table provide supplementary information relating to the profiles viewed by the user such as the profile type (private profile or page), description (if any), URLs to the profiles, cover photo metadata (i.e., photo IDs, sizes, URLs, titles, and creation times for the cover photos), number of mutual friends associated with the profiles (if any), whether a friend request can be sent to the profiles, and the user has liked the page or is a subscriber.

4.4 Conversations and Transferred Files

Facebook allows users to transfer files up to 15MB. When a file is uploaded using the chat window, it will be attached alongside the line of chat messages (if any) and appear as a download link. The sender is allowed to abort a transfer part way through the process. The downloaded files were saved under %Downloads% by default, all of which were given an Alternate Data Stream (ADS) ZoneTransfer marker (ZoneID) with reading ‘ZoneID = 3’, indicating that the files were downloaded from an Internet zone [62]. This also suggests that when a user downloads a file using the Facebook app, there will be records remaining in Windows system files such as $LogFile, $MFT, and $UsnJrnl to indicate the filenames, directory paths, and timestamps for the downloaded files; an excerpt of the $LogFile entries (recovered from the suspect’s workstation) is shown in Fig 6. Analysis of the thumbnail caches stored within %

![Fig 3. The 'pslist' output for the Facebook app.](doi:10.1371/journal.pone.0150300.g003)

![Fig 4. Login records located in the 'analytics_logs' table of Analytics.sqlite database.](doi:10.1371/journal.pone.0150300.g004)
Windows Instant Messaging App Forensics

AppData%\Local\Packages\Package ID\AC\NetCache<\Cache ID>\ and %AppData%\Local\Microsoft\Windows\Explorer\ (henceforth thumbcache) determined that copies of the transferred or downloaded can be recovered. This creates potential for alternative methods for recovery of the deleted files, but the results may not be definitive.

Examinations of the cache databases determined that artefacts of the conversations could be recovered from the Analytics.sqlite and Messages.sqlite databases. Within the 'analytics_logs' table of the former there were timestamp records which reflected the times when the chat tab was turned on, conversations were initiated by the user, as well as files were downloaded. The entry of which could be discerned from the 'name' table column which referenced 'chat_turned_on', 'message_sent_attempt' or 'message_send_state', and 'file_downloaded' respectively. Meanwhile, details about the conversations and file transfers were recovered from the 'messages' table in the latter. Each thread created an entry which comprised the thread ID, conversation texts (if any), UID and username of the sender and the receiver, a count of the number of times the message was sent, file attachment metadata (i.e., sender’s username and ID as well as filename, file size, and format references for the files transferred as shown in Fig 7), and other relevant information as shown in Fig 8. Additionally, the ‘users’ table (of the Messages.sqlite database) could provide additional information pertaining to the correspondents including the UIDs, email addresses, Facebook names, last active times and other information as detailed in Fig 9.

Undertaking data carving of the memory captures and unallocated space only produced matches to the transferred/downloaded sample files. By searching for terms unique to the app cache databases (i.e., table column names), it was possible to recover complete/partial fragments of the databases in plain text (similar to other IM scenarios). However, there was no common footer information to indicate the file structure. Fig 10 illustrates that records of conversations from the 'messages' table (of Messages.sqlite database) can be located using the table column name 'm_mid'. Moreover, we were also able to locate copies of Asynchronous JavaScript and XML (AJAX) objects for the Facebook chat in the memory captures. The

---

**Fig 5. The ‘friends’ table of Friends.sqlite database.**

doi:10.1371/journal.pone.0150300.g005

<table>
<thead>
<tr>
<th>Uid</th>
<th>Name</th>
<th>First_Name</th>
<th>Middle_Name</th>
<th>Last_Name</th>
<th>Contact_Email</th>
<th>Phone</th>
<th>Profile_Uri</th>
<th>Is Pushable</th>
<th>Has Messenger</th>
<th>Communication_Sent</th>
<th>Birthday_Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Kelvin Sky</td>
<td>Kelvin</td>
<td></td>
<td></td>
<td><a href="mailto:fcccaster@gmail.com">fcccaster@gmail.com</a></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td>0.000945954955...</td>
</tr>
</tbody>
</table>

---

**Fig 6. SLogFile entries for the Facebook app’s file download.**

doi:10.1371/journal.pone.0150300.g006
artefacts could provide a clear indication of contact in Unix epoch format, Facebook usernames and UIDs of the correspondents, and conversation texts as depicted in Fig 11. The JSON coding could be a suitable search keyword for future searches. The presence of the remnants in the memory space of ‘Facebook.exe’ confirmed that the texts were associated with the Facebook app.

Inspecting the network traffic, it was observed that the transferred files were uploaded to IP addresses 31.13.70.*, 31.13.67.*, and 31.13.67.* with URLs referencing ‘upload.facebook.com’. The downloaded files were seen from IP addresses 31.13.70.*, and the URLs were prefixed with ‘cdn.fbsdx.com’. Meanwhile, the IP addresses i.e., 31.13.79.* and 31.13.76.102 were observed in relation to the conversations, with URLs referencing ‘5-edge-chat.facebook.com’—see Table 4 for details. Although the contents were encrypted completely, the IP addresses and URLs highlighted as part of our research may assist a practitioner in scoping the Facebook activities undertaken by a suspect in future investigations. Additionally, the IP addresses can be correlated with the ‘netscan’ output (of Volatility) to obtain information regarding the running process (i.e., PID, process creation time, and socket states) as detailed in Fig 12.

4.5 Real-time Notifications

Facebook notifications prompt users in real-time when activities such as messages and comments were posted on their walls, or wall post tagging took place. Analyses of the directory listings only revealed records of the notifications in the ‘notifications’ table of Notifications.sqlite database. The records contained the senders’ UIDs, notification texts, URLs, update and creation times, whether a notification has been read by the user (‘1’ for read and ‘0’ for unread), and other options useful to aid timeline analysis (see Fig 13).

<table>
<thead>
<tr>
<th>id</th>
<th>thread_id</th>
<th>body</th>
<th>sender</th>
<th>tags</th>
<th>timestamp</th>
<th>seen_time</th>
<th>read</th>
<th>attachments</th>
</tr>
</thead>
<tbody>
<tr>
<td>m_msg.@121</td>
<td>0</td>
<td>Here are some files for you...</td>
<td>&quot;user_id&quot;:1600391123123, &quot;name&quot;:&quot;Kobe Sky&quot;,...</td>
<td>[&quot;inbox&quot;,&quot;read&quot;,&quot;sent&quot;,&quot;source:chat&quot;]</td>
<td>142434705659</td>
<td>14</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>m_msg.@241</td>
<td>0</td>
<td>Here are some files for you...</td>
<td>&quot;user_id&quot;:1600391123123, &quot;name&quot;:&quot;Jackie Sky&quot;,...</td>
<td>[&quot;inbox&quot;,&quot;read&quot;,&quot;sent&quot;,&quot;source:chat&quot;]</td>
<td>142434705659</td>
<td>14</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>m_msg.@121</td>
<td>0</td>
<td>Hello Victim</td>
<td>&quot;user_id&quot;:1600391123123, &quot;name&quot;:&quot;Jackie Sky&quot;,...</td>
<td>[&quot;inbox&quot;,&quot;read&quot;,&quot;sent&quot;,&quot;source:chat&quot;]</td>
<td>142434705659</td>
<td>14</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>m_msg.@241</td>
<td>0</td>
<td>Hello Victim</td>
<td>&quot;user_id&quot;:1600391123123, &quot;name&quot;:&quot;Jackie Sky&quot;,...</td>
<td>[&quot;inbox&quot;,&quot;read&quot;,&quot;sent&quot;,&quot;source:chat&quot;]</td>
<td>142434705659</td>
<td>14</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>m_msg.@121</td>
<td>0</td>
<td>Hello Suspect</td>
<td>&quot;user_id&quot;:1600391123123, &quot;name&quot;:&quot;Kobe Sky&quot;,...</td>
<td>[&quot;inbox&quot;,&quot;read&quot;,&quot;sent&quot;,&quot;source:chat&quot;]</td>
<td>142434705659</td>
<td>14</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig 8. The ‘messages’ table of Messages.sqlite database.
4.6 Uninstallation of the Facebook App

Uninstallation of the Facebook app did not create uninstallation files. When the uninstallation was taken place, only the installation folder remained, but was moved to %Program Files% \WindowsApps\Deleted. Other footprints such as remnants from RAM, unallocated space, and system files such as pagefile.sys, shortcuts, event logs, prefetch files, $LogFile, $MFT, as well as $UsnJrnl were not affected by uninstallation process. The uninstallation also created additional references to the directory paths and timestamp information for the files removed during the uninstallation in $LogFile, $MFT, as well as $UsnJrnl.

5. Analysis of the Skype App

Skype is a popular IM and Voice over Internet Protocol (VoIP) application that provides free IM services, audio and video calls between computers and other mobile devices [63]. With the recent launch of Windows 8.1, Skype is now an integrated Windows service. The most recent version of Skype uses the Super Wideband Audio Codec (SILK) [64]. The overlay peer-to-peer network consists of a combination of ordinary and supernodes [57]. An ordinary node is a typical Skype application that provides the users the ability to place calls and send text messages. The supernode serves as a proxy to relay information between nodes with firewall restrictions and an intermediary to handle authentication and user lookups during logins [57].

In this section, we present results of our investigation of artefacts left behind after the use of the Skype (Windows store) app version 3.1.0.1007 on Windows 8.1, such as installation directory paths, usernames, passwords, text of conversations, transferred or downloaded files, records of video and voice calls, and the associated timestamps.

5.1 Installation of the Skype App

Analysis of the directory listing identified that the package ID could be discerned from 'Micro-
soft.SkypeApp_kzf8qxf38zg5c'. The package ID was then used to correlate the 'InstallTime'
registry entry, Windows Store logs, and event logs to determine the installation and accessed

![Fig 9. The 'users' table of Messages.sqlite database.](doi:10.1371/journal.pone.0150300.g009)

![Fig 10. Portion of the 'messages' table of Messages.sqlite database recovered from the memory space of Facebook.exe.](doi:10.1371/journal.pone.0150300.g010)
times. An inspection of the prefetch files determined that the process name (for the Skype app) was masqueraded with 'WWAHost.exe'—the process name for the Store apps written in JavaScript [35]. As the same process name was located for more than one app of the same type, it was not possible to determine exactly which prefetch file was associated with the Skype app.

Table 4. Network information observed for the Facebook app.

<table>
<thead>
<tr>
<th>Registered owner</th>
<th>IP address(es)</th>
<th>URL(s) observed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Akamai Technologies Inc.</td>
<td>23.58.43.27</td>
<td>e8218.ce.akamaiedge.net, ocsp.ws.symphantec.com.edgekey.net, gtssl-ocsp.geotrust.com, g.symcd.com</td>
</tr>
<tr>
<td>Akamai Technologies Inc.</td>
<td>23.62.109.216</td>
<td>a2047.dsp1.akamai.net, fbcdn-profile-a.akamaihd.net</td>
</tr>
<tr>
<td>Akamai Technologies Inc.</td>
<td>23.62.109.87</td>
<td>a591.dspda2.akamai.net, fbcdn-vthumb-a.akamaihd.net.edgesuite.net</td>
</tr>
<tr>
<td>Facebook USA</td>
<td>31.13.70.1</td>
<td>star.c10r.facebook.com, api.facebook.com, <a href="http://www.facebook.com">www.facebook.com</a>, star.facebook.com, upload.facebook.com</td>
</tr>
<tr>
<td>Facebook USA</td>
<td>31.13.70.7</td>
<td>scontent.xx.fbcdn.net, cdn.fbcbx.com</td>
</tr>
<tr>
<td>Facebook USA</td>
<td>31.13.76.102</td>
<td>star.c10r.facebook.com, 5-edge-chat.facebook.com</td>
</tr>
<tr>
<td>DiGi Telecommunications Sdn Bhd</td>
<td>115.164.141.10</td>
<td>a1005.dspw42.akamai.net, fbcdn-sphotos-e-a.akamaihd.net.edgesuite.net, a1005. dsw42.akamai.net, fbcdn-sphotos-e-a.akamaihd.net.edgesuite.net</td>
</tr>
<tr>
<td>DiGi Telecommunications Sdn Bhd</td>
<td>115.164.141.16, 115.164.141.17</td>
<td>a1406.dspw42.akamai.net, fbcdn-sphotos-f-a.akamaihd.net.edgesuite.net, a1406. dsw42.akamai.net, fbcdn-sphotos-f-a.akamaihd.net.edgesuite.net</td>
</tr>
<tr>
<td>DiGi Telecommunications Sdn Bhd</td>
<td>115.164.141.32, 115.164.141.34, 115.164.141.40</td>
<td>a1003.dspw41.akamai.net, fbcdn-sphotos-c-a.akamaihd.net.edgesuite.net, a1404. dspw41.akamai.net, fbcdn-sphotos-d-a.akamaihd.net.edgesuite.net, a1408.dspw43.akamai.net, fbcdn-sphotos-h-a.akamaihd.net.edgesuite.net</td>
</tr>
<tr>
<td>Facebook Inc.</td>
<td>173.252.103.16</td>
<td>orcart.vvv.facebook.com, orcart.facebook.com</td>
</tr>
<tr>
<td>Facebook Inc.</td>
<td>173.252.120.6</td>
<td><a href="http://www.facebook.com">www.facebook.com</a></td>
</tr>
</tbody>
</table>

doi:10.1371/journal.pone.0150300.t004
5.2 Logins

The crucial artefacts were predominantly located in the user-specific %AppData%\Local\Packages\Microsoft.SkypeApp_kzf8qsxf38xg5c\LocalState\.<Skype name>\main.db database (unless otherwise stated, all tables will henceforth be referred to this database). Of particular interest with respect to the logins is the ‘Accounts’ table, which maintains a list of details about the Skype accounts logged in from the computer under investigation. The details comprise the account registration times in Unix epoch format, Microsoft Live usernames, Skype names, users’ full name, birth dates, gender, registered locations, phone numbers, email addresses, homepage URLs (if any), mood texts and the creation times, time zones, and other information useful for user profiling. To recover the avatars used by the users, the practitioner can access %AppData%\Local\Packages\Microsoft.SkypeApp_kzf8qsxf38xg5c\LocalState\avatars\.

Analysis of the Internet Explorer’s web browsing history was able to identify two URLs associated with the logins, which were login.skype.com/login?message=signin_continue&return_url=... and login.skype.com/login/sso?nonce=...). The web browsing history can provide an estimate of the number of times a suspect had accessed Skype as well as the corresponding login times on the computer under investigation.

Examination of the %AppData%\Local\Packages\Microsoft.SkypeApp_kzf8qsxf38xg5c\LocalState\shared.xml file indicated the Skype name and node ID of the user in the ‘Default’ and ‘NodeID’ tags, respectively. The Skype name can prove useful for correlating events initiated by the user during further analysis. Meanwhile, it was observed that the ‘HostCache’ tag maintains a string of the supernode IP addresses and port pairs that Skype builds and refreshes regularly [57]. Each of which is recorded in twelve character hexadecimal strings and prefixed with 0400050041050200... [65]. The shared.xml file also held records of the last used external IP address, port number, and last connected supernode IP address and port pair in the ‘LastIP’, ‘ListeningPort’, ‘Supernode’ tags in decimal format, respectively—see Fig 14; useful to support network analysis.

<table>
<thead>
<tr>
<th>Offset(P)</th>
<th>Proto</th>
<th>Local Address</th>
<th>Foreign Address</th>
<th>State</th>
<th>Pid</th>
<th>Owner</th>
<th>Created</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x7e727061</td>
<td>TCPv4</td>
<td>1.23.45.67:89</td>
<td>2.34.56.78:90</td>
<td>ESTABLISHED</td>
<td>3400</td>
<td>Facebook.exe</td>
<td>2015-01-19 16:29:21 UTC=0000</td>
</tr>
<tr>
<td>0x7e727061</td>
<td>TCPv4</td>
<td>1.23.45.67:89</td>
<td>2.34.56.78:90</td>
<td>ESTABLISHED</td>
<td>3400</td>
<td>Facebook.exe</td>
<td>2015-01-19 16:29:21 UTC=0000</td>
</tr>
<tr>
<td>0x7e727061</td>
<td>TCPv4</td>
<td>1.23.45.67:89</td>
<td>2.34.56.78:90</td>
<td>ESTABLISHED</td>
<td>3400</td>
<td>Facebook.exe</td>
<td>2015-01-19 16:29:21 UTC=0000</td>
</tr>
<tr>
<td>0x7e727061</td>
<td>TCPv4</td>
<td>1.23.45.67:89</td>
<td>2.34.56.78:90</td>
<td>ESTABLISHED</td>
<td>3400</td>
<td>Facebook.exe</td>
<td>2015-01-19 16:29:21 UTC=0000</td>
</tr>
</tbody>
</table>

**Fig 12. The ‘netscan’ output for the Facebook app.**

doi:10.1371/journal.pone.0150300.g012

**Fig 13. The ‘notifications’ table of Notifications.sqlite database.**

doi:10.1371/journal.pone.0150300.g013
Although the process name was masqueraded with 'WWAHost.exe', we could correlate the supernode IP addresses (obtained from the shared.xml file) with the 'netscan' output (of Volatility) to determine the PID. For example, when we mapped the supernode IP address of '111.221.77.148' with the 'netscan' output recovered from our research (see Fig 15), we obtained the PID '656'. The PID could then be used to map the 'pslist' output (of Volatility) to obtain additional information such as the PPID and process creation time as shown in Fig 16.

Further analysis of the unstructured datasets identified that the config.xml and shared.xml files can be potentially carved from the memory dump and unallocated space using the header and footer values of "3C 3F 78 6D 6C 20 76 65 72 73 69 6F 6E 3D 22...3C 2F 3F 78 6D 6C 20 76 65 72 73 69 6F 6E 3D 22...3C 2F 3C 69 62 3E 0D 0A 3C 2F 63 6F 6E 66 69 67 3E 0D 0A...3C 2F 3C 69 62 3E 0D 0A 3C 2F 63 6F 6E 66 69 67 3E 0D 0A...3C 2F 3C 69 62 3E 0D 0A 3C 2F 63 6F 6E 66 69 67 3E 0D 0A" respectively, but the findings may be subject to software updates.

Upon launching the app, it was observed that the host first established a session with Edge-Cast Networks to download Microsoft's certificate revocation list (CRL) on port 80. The next session was established with the Akamai servers to retrieve the contact (i.e., IP address 23.58.236.138) and advertisement information (i.e., IP address 23.58.154.154) on port 443. Then, a session was established with the Microsoft servers (i.e., IP addresses 168.63.212.78 and 137.116.32.77 on port 443) for the traffic management service. When the logins occurred, the host first established several TCP sessions with random supernodes, which we hypothesised for user lookups [57]. Similar to the observation of Azab et al. [57], the IP addresses were associated with a combination of random and destined (33033) port numbers. The next servers accessed were the Windows Live Messenger server (i.e., IP address 65.54.184.60), Windows...
Live servers (i.e., IP addresses 65.55.246.12/C3), as well as Hotmail server (i.e., IP address 65.55.68.104) on port 443 for login authentication and buddy list retrieval. The sessions were subsequently seen with random IP addresses on random UDP ports. Also observed were many connections to the IP addresses 91.190.216.* (referencing 'rstwh.skype-cr.akadns.net' and '1007.0.1.3.9.rst15.r.skype.net') on random TCP port numbers, but we were unable to identify the actual functions of the IP addresses due to lack of information from the URLs as well as encrypted traffic—see Table 5 for details of the captured network traffic. Rebuilding the network files using Netminer, we only recovered certificates that were used to authenticate the HTTPS sites as well as HTML documents and image files from the HTTP sites. Since the network traffic was encrypted (HTTPS), no credential information was recovered from the network captures.

5.3 Contacts

Artefacts of the contacts were located in the ‘Contacts’ table. The artefacts comprised the Skype names, full names, birth dates, gender details, languages, registered locations, contact numbers, email addresses, homepage URLs (if any), mood texts, time zones, last online times, display names, last accessed times, and other information as depicted in Fig 17. Examination of the %AppData%\Local\Packages\Microsoft.SkypeApp_kzf8qxf38zg5c\LocalState\<Skype name>\config.xml file revealed the user ID for the contact with whom the user last communicated as well as the last accessed time. Each contact formed an opening and closing subtag in the ‘u’ tag as shown in Fig 18.
When the Skype account was synced with the Microsoft account, additional profile information was recovered for the contacts in the address book located at %Appdata%\Local\Packages\microsoft.windowscommunicationsapps_8wekyb3d8bbwe\LocalState\Indexed\LiveComm\6e49dfb76dd9b\1207120049\People\AddressBook\26000001_bef42d234ebd42.appcontent-ms. Each contact formed an opening and closing ‘properties’ tag to house the search properties such as search keywords, full names, home addresses, birth dates, phone numbers, and other information as detailed in Fig 19, which may be of value for user profiling. Additionally, the similar information could be located for the user in the %Appdata%\Local\Packages\microsoft.windowscommunicationsapps_8wekyb3d8bbwe\LocalState\Indexed\LiveComm\6e49dfb76dd9b\120712-0049\People\Me\24000001_7b20c4c2b2382.appcontent-ms file.

5.4 IM Conversations and Transferred Files

Examinations of the directory listings determined that the files downloaded were saved in %Downloads%\Microsoft\SkypeApp_kzf8qpxf38g5c\App\ and %AppData%\Local\Packages\Microsoft.SkypeApp_kzf8qpxf38g5c\Local\Skype name\\ReceiveStorage, by default; each of which was given an ADS ZoneID with reading ‘ZoneID = 3’. Meanwhile, copies of the transferred files were located in %AppData%\Local\Packages\Microsoft.SkypeApp_kzf8qpxf38g5c\Local\Skype name\\SendingStorage. The files retained the original filenames and extensions. In addition to the file download or transfer directory paths, we were able to recover copies of thumbnail images for the transferred or downloaded files within the Windows thumbcache.

An inspection of the registry entries observed that each transferred or downloaded file created a Globally Unique Identifier (GUID) key in HKEY_USERS\<SID>\Software\Classes\LocalSettings\Software\Microsoft\Windows\CurrentVersion\AppModel\SystemAppData\Microsoft.SkypeApp_kzf8qpxf38g5c\PersistedStorageItemTable\ManagedByApp. The entries

```xml
<?xml version="1.0"?>
<config version="1.0" serial="78" timestamp="1421686251.63">
    ...
    <Account>
        <LastPartnerId>830</LastPartnerId>
        <LastUsed>1421679670</LastUsed>
        <LocalData>4215</LocalData>
        <Migration>63</Migration>
        <OMigration>1</OMigration>
    </Account>
    ...
    <ui>
        <echo123>9db4df93:2</echo123>
        <harold.2Ecornwall1>4857bb98:2</harold.2Ecornwall1>
    </ui>
    ...
</config>
```

Fig 17. An excerpt of the ‘Contacts’ table of main.db database.

Fig 18. Portion of config.xml file.
of particular interest with the key are 'FilePath' and 'LastUpdatedTime', which hold the directory path and last modified time for the file. When the sample files were opened, references were found for the directory paths and last accessed times in the 'RecentDocs' registry key and 'DLLHOST.EXE.pf' prefetch file.

An inspection of the main.db database located further details regarding the file transfer or download in the 'Transfers' table. The details included the senders' names, transfer types (where 1 indicates receiving and 2 indicates transferring), reasons for transfer failure (if any), storage paths, the times when the transfers were accepted, started and finished, as well as other file transfer information as shown in Fig 20. Records specific to the conversation or file transfer threads were located in the 'Messages' table, which encompassed the senders' Skype names (authors), whether the correspondents were the user's permanent contacts, the times when the threads were sent in Unix epoch format, the message sending status and types (as indicated in Table 6), reasons for message sending failure (if any), and other information as shown in Fig 21. The group chat could be discerned from the 'participant_count' table column given the value higher than 2. Moreover, it was also possible to recover the conversation texts and
metadata associated with the downloaded or transferred files in the 'body_xml' table column (of the 'Messages' table). As can be seen in Fig 22, each downloaded or transferred file forms an opening and closing XML subtag (in the 'files' tag) to record its file size, transfer index, transfer ID, and filename in the 'body_xml' table column.

Another file of forensic interest that will potentially allow a practitioner to recover the conversation history is the 'Chatsync' file located in %AppData%/Local/Packages\Microsoft.SkypeApp_kzf8qxf538zy5c\LocalState\Skype name\Chatsync. The 'Chatsync' file is stored in the format of <Random sixteen character strings>.DAT and is mainly used to facilitate chat log synchronisation between devices [67]. The 'Chatsync' file is chat-session-specific in the sense that a chatsync file is generally created for each chat session. Fig 23 illustrates that the 'Chatsync' files may provide the conversation texts and timestamp information for the chat sessions associated with the Skype user.

Table 6. Details of the 'Messages' table [66].

<table>
<thead>
<tr>
<th>Table field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>4</td>
<td>Conference</td>
</tr>
<tr>
<td></td>
<td>30/39</td>
<td>Video session started/ended</td>
</tr>
<tr>
<td></td>
<td>50/51</td>
<td>Contact ask/permission</td>
</tr>
<tr>
<td></td>
<td>53</td>
<td>Blocked</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>Sent emoticon</td>
</tr>
<tr>
<td></td>
<td>61</td>
<td>Sent text message</td>
</tr>
<tr>
<td></td>
<td>63</td>
<td>Sent contact details</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>Sent SMS</td>
</tr>
<tr>
<td></td>
<td>67</td>
<td>Sent voice message</td>
</tr>
<tr>
<td></td>
<td>68</td>
<td>Sent file</td>
</tr>
<tr>
<td></td>
<td>110</td>
<td>Date of birth</td>
</tr>
<tr>
<td>chatmsg_type</td>
<td>3</td>
<td>Text message</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Group chat</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Data transfer</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>Added contact</td>
</tr>
<tr>
<td>chatmsg_status</td>
<td>2</td>
<td>Text message sent</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Text message downloaded</td>
</tr>
</tbody>
</table>

doi:10.1371/journal.pone.0150300.I006

Fig 21. The 'Messages' table of main.db database.

doi:10.1371/journal.pone.0150300.g021
Unsurprisingly, a manual search for terms unique to the Enron sample files (i.e., ‘pensive’ and ‘parakeet’) as well as table column names of the main.db database produced matches to the plain text copies of the transferred/downloaded files and main.db database in the unstructured datasets, respectively. However, there was no common footer information that could enable future carving of the main.db database. We also located fragments of the payloads for the conversation threads in the memory dump, which held the conversation times, senders and receivers’ Skype names, and conversation texts as highlighted in Fig 24. When file transfers occurred, additional entries were observed for the filenames, file sizes, and file transfer IDs in the payload. The header fields could be suitable search terms for the remnants; a Yarascan search would attribute the remnants to the Skype’s process.

Examination of the network traffic observed that the host established a direct UDP connection with the correspondents during conversations and file transfers, and hence the IP addresses could be detected. However, there was no definitive port number or URL which could enable future identification of the traffic. Further analysis of the network packets determined that the data were fully encrypted, but we were able to estimate when the conversations were taken place from the corresponding timestamp information.

5.5 Voice and Video Calls

Skype allows users to perform voice calls via the free Skype to Skype calls and in the premium version, users could make Skype to mobile or landline calls using Skype credit. In order to enhance the user’s interactive experience, Skype allows users to share free video calls with anyone who has Skype and a webcam or compatible smartphone.

![Fig 22. File transfer metadata recovered from the ‘body_xml’ table column of the ‘Messages’ table.]( doi:10.1371/journal.pone.0150300.g022)

![Fig 23. Portion of the output from Skype Chatsync Reader.]( doi:10.1371/journal.pone.0150300.g023)
Examinations of the directory listings determined that the Skype app does not save the voice and video calls. However, we were able to recover a wealth of caches relating to the voice and video calls in the main.db database. Recalling the ‘Messages’ table, it was observed that entries of the voice or video calls could be differentiated from the ‘type’ table column given the value 30, 39, or 67 (see Table 6). Details of the voice or video calls were recovered from the ‘Calls’ table, which comprised the callers’ Skype names, the times when the calls were started, the call durations in seconds, and whether the calls were incoming calls, conference calls, and put on hold—see Fig 25. Additionally, the ‘CallMembers’ table provided additional information associated with the contacts with whom the user had voice or video calls such as the Skype names, full names, call charges, reasons for call failures (if any), graphical user IDs (represented in <User’s Skype name>-<Correspondent’s Skype name>-<Call name>), external IP addresses of the correspondents, call statuses, the times when the calls were started, the call durations, whether the calls were incoming or outgoing, conference calls, and from permanent contacts.

Examinations of the network traffic of the voice and video calls observed that the app established a session with the CloudFlare (GlobalSign) server for Online Certificate Status Protocol (OSCP) stapling and with the Verisign server for certificate authentication. When the calls occurred, the IP addresses were allocated to the supernodes (on random TCP ports) and then to the Windows Live server (i.e., IP address 65.55.246.85) on port 443, which we theorised for user lookups and authentications. The network traffic was subsequently seen with random IP addresses and UDP ports, which were hypothesised from supernodes responsible for bridging the VoIP, but the contents were encrypted completely.

![Fig 24. Remnants of Skype’s payload header recovered from RAM.](https://doi.org/10.1371/journal.pone.0150300.g024)

![Fig 25. An excerpt of the ‘Calls’ table of main.db database.](https://doi.org/10.1371/journal.pone.0150300.g025)
5.6 Video Messages

Skype allows the users to share video messages (video recordings) with other online and offline users. The video messages are sent as a link in Skype version 6.5 or older, which requires a secret code access.

Sending a video message, it was observed that the Skype app stored a copy of the video message in %AppData%\Local\Packages\Microsoft.SkypeApp_kzf8qxf38zg5c\LocalState\<Skype name>\media of the sender’s device by default. The video message also created a thumbnail image in %AppData%\Local\Packages\Microsoft.SkypeApp_kzf8qxf38zg5c\LocalState\<Skype name>\thumbnails.

Analysis of the main.db database revealed that the Skype app cached notifications of the video messages in the 'body_xml' table column of the 'Messages' table, and the entry of which could be discerned from the XML tag 'videomessage'. The notification records provided the video message IDs, public links, and secret codes (sent from Skype application version 6.5 or older) for the video messages sent or received by the user as highlighted in Fig 26. Meanwhile, details of the video messages sent/received could be located in the 'VideoMessages' table, which included the directory paths, public links, titles, descriptions (if any), author names, creation times, transferring or receiving times as illustrated in Fig 27.

5.7 Uninstallation of the Skype App

Uninstallation of the Skype app did not remove the installation folders like as was presented for the Facebook app. However, the application folder was removed from the file system completely. Analysis of the unallocated space, RAM, as well as a variety Windows system files (i.e., $LogFile, $MFT, $UsnJrnl, pagefile.sys, shortcuts, event logs, prefetch files, and thumbcache files) resulted in the recovery of artefacts created prior to uninstallation of the app, with additional references to the directory paths and timestamp information for the files removed during the uninstallation in $LogFile, $MFT, $UsnJrnl.

6. Discussion

In this research, we identified artefacts common to investigating the Windows Store apps for IM. Previous studies only addressed dead analysis of the IM apps, while we focus on both the volatile and non-volatile artefacts. Our experiments showed that the Facebook and Skype apps maintain a wealth of caches of forensic interest within the 'localstate' application folder in Sqlite database unencrypted, which seem to agree with the findings of Lee and Chung [34]. This indicated that when a user has used a Windows Store app for IM, there will be records remaining in the application folder to support reconstruction of the logins, contact lists, conversations, file transfers, and other relevant IM activities, assuming that the app is not removed.

![Fig 26. Video message metadata recovered from the 'body_xml' table column of the 'Messages' table.](https://doi.org/10.1371/journal.pone.0150300.g026)

![Fig 27. An excerpt of the 'VideoMessages' table of main.db database.](https://doi.org/10.1371/journal.pone.0150300.g027)
Although several registry keys new to the Windows Store apps could be recovered, it was determined that the Windows Store apps record significantly less information of interest to IM forensics in comparison to traditional client desktop application. While artefacts of the user profiles, contact lists and recent communications could be potentially recovered from the registry of the older Windows IM client applications [16, 21, 36–38, 42, 43], only installation metadata (i.e., install paths and times) could be recovered for the Windows Store apps, albeit records of the transferred files could be recovered in some cases. This is likely resulted from the adoption of the app caches. Similar to any other Windows client applications, our examinations of the system files such as $LogFile, $MFT, $UsnJrnl, shortcuts, event logs, thumbnail cache, as well as the ‘recentdocs’ registry key revealed that additional timestamp information could be recovered to support evidence found in all scenarios, but results may not be definitive.

It should be noted, however, that the significance, amount, and location of artefacts could vary in accordance to the Windows Store apps under investigation. For instance, in our research, it was determined that:

- both the Facebook and Skype apps maintain a different directory structure in the application folders;
- the apps hold different database schema for the application caches;
- caches of the Facebook app appear only when the user is logged in from the app, while caches of the Skype app remain resident throughout the lifetime of the app;
- the Skype app caches copies of the transferred and downloaded files in the application folder but this is not the case with the Facebook app;
- only the Skype app holds records of the transferred or downloaded files in HKEY_USERS \<SID>\Software\Classes\LocalSettings\Software\Microsoft\Windows\CurrentVersion\App-Model\SystemAppData\<Package ID>\PersistedStorageItemTable\PersistedByApp.

The findings suggested that while a method can be generally defined to guide the investigation of the Windows Store apps, a different process may be necessary for investigating the different IM apps.

Our examinations of the physical memory captures indicated that the memory dumps can provide a potential alternative method for recovery of the application caches in plain text, with the exception of the login password. The fact that there was no clear text password in the hard drives and memory dumps should perhaps be unsurprising since the credential information is securely encrypted in the Credential Locker [29]. Nevertheless, a practitioner must keep in mind that memory changes frequently according to users’ activities and will be wiped as soon as the system is shut down.

In some cases, remnants of the caches could be located in the swap file (pagefile.sys) and unallocated space. The most likely explanation for the remnants is that the system swapped inactive memory pages containing the application caches out of the memory to the hard disk during the system’s normal operation. As the remnants were recovered with minimal space configuration in our research, we believe there will be a greater chance of remnants on a typically larger system. Although the network traffic was encrypted, sufficient IP address and URL references could be located for scoping the user activities as well as requesting for assistance from counterparts overseas (i.e., via Interpol). Hence, we recommend that the physical memory and network captures should be undertaken wherever practical. Table 7 summarises the key artefacts located as part of our research.
## Table 7. Summary of findings.

<table>
<thead>
<tr>
<th>Source of evidence</th>
<th>Facebook app</th>
<th>Skype app</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registry branches of forensic interest.</td>
<td>HKEY_USERS&lt;SID&gt;\Software\Classes\Local Settings \Software\Microsoft\Windows\CurrentVersion\AppModel\Repository\Families\Facebook\Facebook_1.4.0.9_x64_8xx8rvfyw5nnt</td>
<td>HKEY_USERS&lt;SID&gt;\Software\Classes\Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\Repository\Families\Microsoft\SkypeApp\kzf8qx38zg5c\Microsoft. SkypeApp_2.0.0.5011_x86_kzf8qx38zg5c</td>
</tr>
<tr>
<td>Directory paths/ files of forensic interest</td>
<td>%AppData%\Local\Temp\winstore.log</td>
<td>%AppData%\Local\Temp\winstore.log</td>
</tr>
<tr>
<td></td>
<td>%AppData%\Local\Packages\winstore\cw5n1h2txyewy\AC\Temp\winstore.log</td>
<td>%AppData%\Local\Packages\winstore\cw5n1h2txyewy\AC\Temp\winstore.log</td>
</tr>
<tr>
<td></td>
<td>Analytics.sqlite, FriendRequest.sqlite, Friends. sqlite, Messages.sqlite, Notifications.sqlite, and Stories. sqlite databases stored in %AppData%\Local\Packages\Facebook\Facebook_8xx8rvfyw5nnt\LocalState\&lt;User-specific Facebook ID&gt;\DB\</td>
<td>User-specific %AppData%\Local\Packages\Microsoft. SkypeApp\kzf8qx38zg5c\LocalState\&lt;User name&gt;!main.db</td>
</tr>
<tr>
<td></td>
<td>Caches of the downloaded files stored in %AppData%\Local\Packages\Package ID\AC\NetCache\Cache ID\</td>
<td>%AppData%\Local\Packages\Microsoft. SkypeApp\kzf8qx38zg5c\LocalState\&lt;User name&gt;\shared.xml</td>
</tr>
<tr>
<td></td>
<td>%AppData%\Local\Packages\Microsoft. SkypeApp\kzf8qx38zg5c\LocalState\&lt;User name&gt;\ChatSync\</td>
<td>%AppData%\Local\Packages\Microsoft. SkypeApp\kzf8qx38zg5c\LocalState\&lt;User name&gt;\ChatSync\</td>
</tr>
<tr>
<td></td>
<td>%AppData%\Local\Packages\Microsoft. SkypeApp\kzf8qx38zg5c\LocalState\&lt;User name&gt;\avatars\</td>
<td>%AppData%\Local\Packages\Microsoft. SkypeApp\kzf8qx38zg5c\LocalState\&lt;User name&gt;\avatars\</td>
</tr>
<tr>
<td></td>
<td>%Downloads%\Microsoft. SkypeApp\kzf8qx38zg5c\App\</td>
<td>%Downloads%\Microsoft. SkypeApp\kzf8qx38zg5c\App\</td>
</tr>
<tr>
<td></td>
<td>%AppData%\Local\Packages\Microsoft. SkypeApp\kzf8qx38zg5c\LocalState\&lt;User name&gt;\ReceiveStorage\</td>
<td>%AppData%\Local\Packages\Microsoft. SkypeApp\kzf8qx38zg5c\LocalState\&lt;User name&gt;\ReceiveStorage\</td>
</tr>
<tr>
<td></td>
<td>APPCONTENT-MS files located in %Appdata%\Local\Packages\windowscommunicationsapps_8wekyb39dbbwechatProfilepictures of the user and the contacts</td>
<td>APPCONTENT-MS files located in %Appdata%\Local\Packages\windowscommunicationsapps_8wekyb39dbbwe</td>
</tr>
<tr>
<td></td>
<td>%\Appdata%\Local\Packages\microsoft.windowscommunicationsapps_8wekyb39dbbwelocalState\</td>
<td>%\Appdata%\Local\Packages\microsoft.windowscommunicationsapps_8wekyb39dbbwelocalState\</td>
</tr>
<tr>
<td></td>
<td>IndexedLiveComm\16e4f9df0f076dd9b120712-0049</td>
<td>IndexedLiveComm\16e4f9df0f076dd9b120712-0049</td>
</tr>
<tr>
<td></td>
<td>People\AddressBook\</td>
<td>People\AddressBook\</td>
</tr>
<tr>
<td></td>
<td>%\Appdata%\Local\Packages\microsoft.windowscommunicationsapps_s_8wekyb39dbbwe\LocalState\IndexedLiveComm\16e4f9df0f076dd9b120712-0049</td>
<td>%\Appdata%\Local\Packages\microsoft.windowscommunicationsapps_s_8wekyb39dbbwe\LocalState\IndexedLiveComm\16e4f9df0f076dd9b120712-0049</td>
</tr>
<tr>
<td></td>
<td>Peoples\W\Me\</td>
<td>Peoples\W\Me\</td>
</tr>
<tr>
<td>Prefetch files</td>
<td>FACEBOOK.EXE.pf</td>
<td>WWAHOST.EXE.pf</td>
</tr>
<tr>
<td>Link files</td>
<td>Located link files for the transferred or downloaded files in %\AppData\Roaming\Microsoft\Windows\Recent\</td>
<td>Located link files for the login page as well as the transferred or downloaded files in %\AppData\Roaming\Microsoft\Windows\Recent\</td>
</tr>
<tr>
<td>Thumbcache files</td>
<td>Thumbnail images for the transferred or downloaded files</td>
<td>Thumbnail images for the transferred or downloaded files</td>
</tr>
<tr>
<td>Swap files and physical memory dumps</td>
<td>Copies of the files of forensic interest as well as transferred or downloaded files unencrypted</td>
<td>Copies of the files of forensic interest as well as transferred or downloaded files in plain text</td>
</tr>
<tr>
<td>Unallocated space</td>
<td>Filename and path references for the files of forensic interest and transferred or downloaded files</td>
<td>Filename and path references for the files of forensic interest and transferred or downloaded files</td>
</tr>
<tr>
<td></td>
<td>The process name could be discerned from ‘Facebook.exe’</td>
<td>The process name could be discerned from ‘WWAHost.exe’</td>
</tr>
<tr>
<td></td>
<td>Copies of the files of forensic interest as well as transferred or downloaded files in plain text</td>
<td>Copies of the files of forensic interest as well as transferred or downloaded files in plain text</td>
</tr>
</tbody>
</table>

(Continued)
7. Conclusion and Future Work

Instant messaging (IM), such as VoIP apps, are increasingly popular among individuals and business organisations [68], including criminals. To ensure the most effective collection of evidence of relevance, it is important that a practitioner possess an up-to-date understanding of different technologies [69–77]. This paper presented the findings from our forensic examination (acquisition and reconstruction of the terrestrial artefacts left by the use) of two popular Windows Store IM apps, namely Facebook and Skype. The study consisted of installation, uninstallation, logins, conversations, transferred files, and other IM activities specific to the apps investigated.

The results indicated that use of the Windows Store apps IM apps can leave behind incriminating evidential material useful or critical to an investigation on the hard drive, memory dumps, and network captures. The artefacts located as part of our experiments are likely to be common with other Windows Store IM apps as well as newer Windows OS (i.e., Windows 10), since the apps share a common feature set. While the implementation may vary between different IM apps, we contended that practitioners could use the artefacts identified in this research as a basis for their investigation of the client as a potential evidence source.

Future work would include:

1. Extending this study to new (version of) apps, including apps popular in other countries (i.e., WeChat and LINE), to have an up-to-date forensic understanding of these technologies that can be used to inform investigations.

2. Proposing a method for analyzing new (as of yet) unknown apps with similar functionality (ies). If such a method can be developed, evaluation might demonstrate that it can be applied to a new app, or even implemented into a tool.

Author Contributions

Conceived and designed the experiments: TYY AD KKRC. Performed the experiments: TYY. Analyzed the data: TYY. Contributed reagents/materials/analysis tools: TYY AD KKRC. Wrote the paper: TYY AD KKRC ZM.

References


mилуэйкенс.org/2014/05/21/special-report-diploma-mill-scams-continue-to-plague-milwaukee-


37. Dickson M. An examination into Yahoo Messenger 7.0 contact identification. Digital Investigation. 2006 Sep; 3(3):159–65


