DigLA – A Digsby Log Analysis Tool to Identify Forensic Artifacts

Muhammad Yasin, Muhammad Abulaish*

Center of Excellence in Information Assurance, King Saud University
Riyadh – 11653, Saudi Arabia
yaseenyns@gmail.com, mAbulaish@ksu.edu.sa

Abstract

Since the inception of Web 2.0, instant messaging, e-mailing, and social networking have emerged as cheap and efficient means of communication over the Web. As a result, a number of communication platforms like Digsby have been developed by various research groups to facilitate access to multiple e-mail, instant messaging, and social networking sites using a single credential. Although such platforms are advantageous for end-users, they present new challenges to digital forensic examiners because of their illegitimate use by anti-social elements. To identify digital artifacts from Digsby log data, an examiner is assumed to have knowledge of the whereabouts of Digsby traces before starting an investigation process. This paper proposes a design for a user-friendly GUI-based forensic tool, `DigLA`, which provides a unified platform for analyzing Digsby log data at different levels of granularity. `DigLA` is also equipped with password decryption methods for both machine-specific and portable installation versions of Digsby. By considering Windows registry and Digsby log files as dynamic sources of evidence, specifically when Digsby has been used to commit a cyber crime, this paper presents a systematic approach to analyzing Digsby log data. It also presents an approach to analyzing RAM and swap files to collect relevant traces, specifically the login credentials of Digsby and IM users. An expected insider attack from a server security perspective is also studied and discussed in this paper.

Keywords: Digital forensics, Digsby log analysis, forensic artifacts, insider attack, RAM analysis.

1. Introduction

Because of the increasing popularity of social media for sharing, communicating, and transferring various types of data over the Web, multi-purpose applications like Digsby that provide access to multiple applications using a single credential are becoming popular. Digsby provides a unified platform which facilitates multiple online activities like instant messaging, e-mailing, and social networking under a common umbrella. According to the `TechCrunch` report[1], Digsby has more than three million registered users. Digsby supports Gmail, Hotmail, Yahoo mail, AOL mail, and POP and IMAP accounts. The current version of Digsby (build 90, r30044) manages numerous instant messaging protocols, including Google Talk, Live Messenger, Yahoo Messenger, AOL Instant Messenger (AIM), ICQ, Jabber, Facebook and MySpace instant messengers. Digsby helps its users stay up-to-date about activities on social and professional networks such as Facebook, MySpace, Twitter, and LinkedIn. Digsby users can use aliases and merge multiple accounts to avoid duplicate contacts. Although such features are advantageous for end-users, they pose challenges when digital crime investigators try to identify the specific account used for an illicit purpose. Moreover, the unified platform provided by Digsby to facilitate access to multiple applications by end users using a single credential poses challenges for investigators trying to identify relevant and reliable traces embedded within its log files stored at different locations.

Recently, a number of research efforts have been directed towards the identification and collection of forensic artifacts from instant messaging log data [1, 2], particularly Yahoo Messenger [3, 4], MSN [5], Windows Live Messenger [6, 7], AOL/AIM [8, 9], Tencent QQ [10, 11], Skype [2, 12], Pidgin [13], and Trillian [14]. In addition, other studies have involved the investigation of Web-based instant messengers that are volatile in nature [15] and the analysis of instant messengers on Apple iPhones [16]. However, to the best of the authors' knowledge, no attempt has been made so far to analyze `Digsby` log data in a unified manner, and consequently no tool has been developed for this purpose.

This paper proposes a design for a user-friendly GUI-based forensic tool, `DigLA` (Digsby Log Analyzer), which can be used by forensic examiners to analyze Digsby log data to identify digital artifacts without knowing their whereabouts in the system. An in-depth analysis of RAM (random-access memory) and swap files to acquire login credentials either in the clear or encrypted is also presented. `DigLA` facilitates the automatic collection of digital traces from different locations in the underlying computer and system files and stores them locally for further analysis or future reference. It supports the collection of digital traces from both machine-specific and portable Digsby installations. The forensic traces identified by `DigLA` involve

*Corresponding author; Senior member: IEEE, ACM, and CSI; E-mail: abulaish@ieee.org; Phone: +966-1-4696468; Fax: +966-1-4696453

[1] `http://techcrunch.com/2011/04/19/tagged-buys-popular-socialinstant-messaging-client-digsby/`

mainly chat logs, login credentials, significant files and folders, and the information embedded within them. All the information collected during analysis is critically examined and presented with necessary details to assist examiners in real-life scenarios. `DigLA` is also integrated with a password decryption module which decrypts encrypted passwords for both machine-specific and portable Digsby installation versions. At present, `DigLA` runs only under the home and professional editions of Windows-XP (SP-2) and Windows 7, but it can be enhanced to adapt to other versions of Windows or other operating systems. Administrator and limited user accounts have been created and used to analyze `DigLA` behavior in different situations. Test scenarios in virtual machine `VMware`[2] have been created, and analyses of both static data and live systems have been performed. `Registrar Lite`[3] and the Windows-provided registry editor *regedit.exe* were used to conduct registry analyses. `WinHex`[4] was used for critical examination of RAM, swap files, and various critical files such as *logininfo.yaml* and *iconhashes.dat*. Finally, uninstall results were gathered and examined to identify associated digital artifacts.

The rest of the paper is organized as follows. After a brief introduction to the Digsby login and authentication process in Section 2, Section 3 presents the design of `DigLA`. This section also presents the features and artifact collection process of `DigLA`. Section 4 provides the implementation details of the password decryption techniques for both machine-specific and portable Digsby installation versions. A discussion of a possible insider attack also constitutes an important part of this section. Section 5 further describes the analysis of RAM and swap files to collect login credentials and contact IDs. Finally, Section 6 presents conclusions.

2. Digsby Login and Authentication Process

Figure 1 presents a step-by-step user login and authentication process for Digsby. Similarly to other instant messengers (IMs) and IM aggregators, registration is a mandatory requirement in Digsby before accessing the underlying services. To register, a user needs to choose an ID, the availability of which is ensured by Digsby through a table lookup process in its local database. Once the selected ID has been confirmed to be available, the user receives a confirmation message indicating successful completion of the registration process. Meanwhile, Digsby generates a hash code for the user password and stores it in a local database. Instead of storing an unencrypted password on the server, Digsby stores the password hash code to enhance user security. After installing Digsby applications, the user supplies login credentials. Digsby checks for application updates and tries to establish a connection with the server for authentication purposes. After successful authentication, the server is synchronized with the requested client, and his/her status is updated to the group members that are online. Finally, the user supplies the login credentials of the instant messengers, email accounts, and social networks that he/she wants to access through the Digsby ID. All supplied passwords are encrypted using the Digsby password and stored in the Digsby database[5]. Thereafter, whenever the user logs in to his/her Digsby account, the login credentials for all protocols are confirmed automatically. In this way, a user can access multiple services through Digsby using a single login and password.

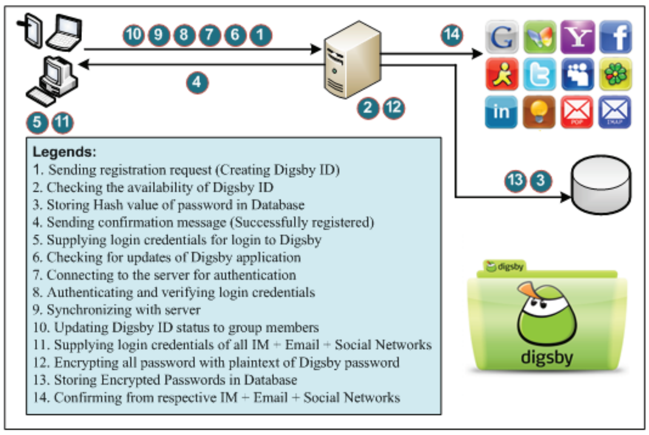

10 9 8 7 6 1
4
5 11
2 12
14
13 3
Legends:
1. Sending registration request (Creating Digsby ID)
2. Checking the availability of Digsby ID
3. Storing Hash value of password in Database
4. Sending confirmation message (Successfully registered)
5. Supplying login credentials for login to Digsby
6. Checking for updates of Digsby application
7. Connecting to the server for authentication
8. Authenticating and verifying login credentials
9. Synchronizing with server
10. Updating Digsby ID status to group members
11. Supplying login credentials of all IM + Email + Social Networks
12. Encrypting all password with plaintext of Digsby password
13. Storing Encrypted Passwords in Database
14. Confirming from respective IM + Email + Social Networks
digsby

Figure 1: Digsby login and authentication process

3. Proposed Digsby Log Analyzer (`DigLA`)

This section presents the design of the proposed Digsby log analysis tool, `DigLA`, which can be used to identify and collate digital artifacts stored by Digsby at different locations for purposes of analysis. Figure 2 provides a summary of the important Digsby directory paths, locations, and files that are examined by `DigLA` to find digital artifacts. `DigLA` provides a user-friendly graphical interface to access digital artifacts stored at various locations. It provides three basic functionalities to examiners to collect traces from the Digsby log repository. First, an investigator can view the Windows registry entries to identify the search page/toolbar, uninstallation path, and most recent usage of Digsby. Second, an investigator can view Digsby log files that contain information related to installation paths, executable files, login credentials, proxy settings, configuration files, last usage time-stamp, aliases or alternate names for contact IDs, exchanged files, separate chat logs for each instant messenger, and pictures associated with contact IDs. A snapshot of the `DigLA` graphical interface showing all this information is presented in Figure 3. Finally, an investigator can decrypt Digsby-encrypted passwords that are collected either from machine-specific installations or from portable installation versions of Digsby.

The authors performed an uninstallation process under both Windows XP and Windows 7 to analyze the relevant artifacts. It should be noted that Digsby does not provide *program utility*

[2]http://www.vmware.com/
[3]http://www.resplendence.com/download/RegistrarLite.exe
[4]http://www.x-ways.net/winhex/index-m.html
[5]http://wiki.digsby.com/doku.php?id=security

and *complete* options during its uninstallation and does not remove any artifacts from Windows directories. Therefore, even though Digsby has been uninstalled from a computer, an examiner can still collect relevant artifacts. However, the examiner may not be able to extract the artifacts from the Windows registry uninstallation key, except for the Digsby usage timestamp, if the investigation starts within a short period of time after the uninstallation process.

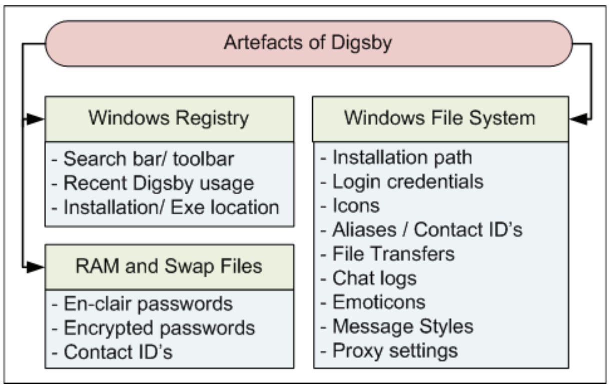


Figure 2: Directory paths, locations, and files containing Digsby artifacts

`DigLA` manages chat logs for each instant messenger as a separate single file which can be analyzed at different levels of granularity using third-party data mining tools to learn about the malicious activities of a suspect based on the semantics of different segments of textual conversations. Graph visualization and link mining techniques can be used to identifying the network of a suspect and the role of the concerned person in it. All these tasks are outside the scope of this paper, but constitute directions for future research to enhance the functionality of `DigLA`. Further details of the `DigLA` forensic artifact identification process are presented in the following subsections.

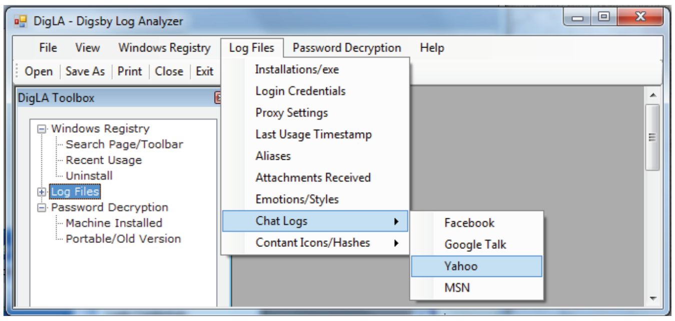
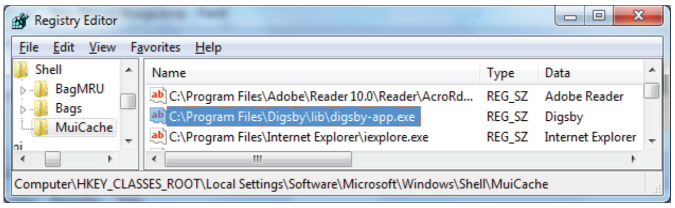


Figure 3: A snapshot of `DigLA` graphical user interface

3.1. Windows Registry Analysis

In contrast to other instant messengers [1, 17, 18] and download managers [19, 20, 21], the Windows registry does not maintain the history of Digsby activities such as instant messages, email messages, and activities performed on social networks. The only information maintained by the Windows registry is the execution path, uninstall location, search bar, and recent usage status of Digsby. During the installation process, the Digsby *search bar* checkbox is checked by default, which is generally the default option chosen by Digsby users. The history of the Digsby search toolbar is maintained under the *HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\SearchScopes\{0633EE93-D776-472f-A0FF-E1416B8B2E3A}* key, which includes the URL address and display name. The *MUICache* key, located at *HKEY_USERS\S-1-5-21-1757981266-1708537768-725345543-500\Software\Microsoft\Windows \ShellNoRoam\MUICache* contains the footprints of Digsby, as shown in Figure 4, which provide clues to the recent usage of Digsby applications. The time-stamp values can be collected using `Registrar Lite`[6] or `FTK Registry Viewer`[7]. The *Uninstall* branch points to all installed applications in a computer. The *Digsby* key contains registry key values, mainly including the execution path, the uninstall path and the publisher, at *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows \Current Version\Uninstall\Digsby*.



Name	Type	Data
C:\Program Files\Adobe\Reader 10.0\Reader\AcroRd...	REG_SZ	Adobe Reader
C:\Program Files\Digsby\lib\digsby-app.exe	REG_SZ	Digsby
C:\Program Files\Internet Explorer\iexplore.exe	REG_SZ	Internet Explorer

Computer\HKEY_CLASSES_ROOT\Local Settings\Software\Microsoft\Windows\Shell\MuiCache

Figure 4: A snapshot of the *MUI Cache* in Windows registry

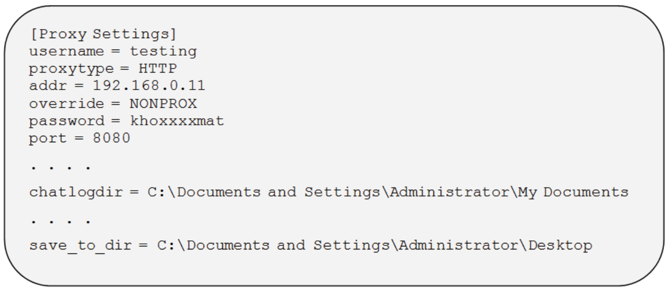
3.2. Digsby Log Data Analysis

Digsby maintains important information under various paths in the Windows file system. Tables 1, 2, and 3 present various types of digital artifacts and their corresponding directory paths in Windows XP, Windows 7, and portable installations, respectively. The artifacts include Digsby login credentials, proxy and configuration settings, emoticons, message styles, and cache and contact icons. Because Digsby does not provide an option to its users to change the location of chat logs, the chat log data can be accessed in their default location, which is very helpful to investigators. Digsby also stores the last login timestamp in the *systemlog.txt* file and maintains all crucial information under a single path, *<Directory path>\Digsby\App*, for portable devices, which assists the forensic investigator in collecting all information from a single location. The following sub-sub-sections present a detailed discussion of these forensic artifacts.

3.2.1. Chat Logs

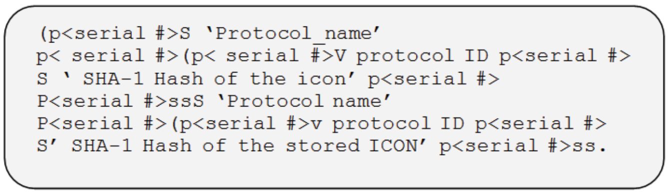
Because Digsby does not implement a hashing or encryption algorithm to secure chat logs, it is fairly easy for an examiner to access them without using a decryption algorithm. Digsby uses HTML format to store chat logs in *My Documents* and maintains them on a daily basis for each contact person in a separate

[6]http://www.resplendence.com/download/RegistrarLite.exe
[7]http://accessdata.com/support/adownloads



```
[Proxy Settings]
username = testing
proxytype = HTTP
addr = 192.168.0.11
override = NONPROX
password = khoxxxxmat
port = 8080
. . . .
chatlogdir = C:\Documents and Settings\Administrator\My Documents
. . . .
save_to_dir = C:\Documents and Settings\Administrator\Desktop
```

(a) *digsbylocal.ini* file containing proxy settings, chat log, and directory paths for received/downloaded files



```
 (p<serial #>S 'Protocol_name'
p< serial #>(p< serial #>V protocol ID p<serial #>
S ' SHA-1 Hash of the icon' p<serial #>
P<serial #>ssS 'Protocol name'
P<serial #>(p<serial #>v protocol ID p<serial #>
S' SHA-1 Hash of the stored ICON' p<serial #>ss.
```

(b) *iconhashes.dat* file format

Figure 5: A snapshot of *digsbylocal.ini* and *iconhashes.dat* files

Table 1: Artifacts and their locations in Windows XP

Artifacts	**Directory path**
Default installation path, executables	C:\Program Files \Digsby C:\Documents and Settings\<Windows_ID>\Local Settings\Application Data\Digsby\App <Directory Path>\Digsby\App
Cache, password, temporary data, icons, last usage time-stamp, aliases	C:\Documents and Settings\<Windows_ID>\Local Settings\Application Data\Digsby
File transfer	C:\Documents and Settings\<Windows_ID>\Desktop
Emoticons, message styles	C:\Documents and Settings\<Windows_ID>\Local Settings\Application Data\Digsby
Chat and update history	C:\Documents and Settings\<Windows_ID>\My Documents\Digsby Logs

folder, the name of which is determined using the *contact ID* of the respective user. A pre-specified file naming convention (*yyyy-mm-dd.html*) is used to create files for maintaining chat logs. This pattern is exploited by `DigLA` to search log data for a particular date on which malicious activity was performed.

In addition, Digsby maintains a history of all supported protocols in separate folders, as shown in Table 4. For example, Facebook chat logs are maintained under *<Digsby ID>\fbchat\<Protocol ID>\571xxxxx3_fbchat*. This folder path provides crucial information such as the sender ID, receiver ID, protocol ID, and date- and time-stamp of the instant messages. It also provides the numeric Facebook ID of the contact user, which can be used to determine the actual identity of the user using search engine APIs. In this way, an examiner can trace all persons with whom a user was in contact through the Facebook chat service.

3.2.2. Proxy Settings

Digsby includes *digsbylocal.ini* file at the *C:\Documents and Settings\<Windows_ID>\Local Settings\Digsby* location, which contains proxy settings, chat log directory location, and location information of downloaded or transferred files, as shown in Figure 5(a). The proxy username and password are stored in plain text. The variable *chatlogdir* contains the complete path of the chat history directory. Similarly, *save_to_dir* maintains the complete path of the received files.

3.2.3. Contact Icons

Digsby maintains an iconhashes.dat file at the *C:\Documents and Settings\<Windows_ID>\Local Settings \Digsby\cache\iconhashes.dat* location, which contains information about all contact IDs and icons of added contacts. The format used to store the relevant information in the *iconhashes.dat* file is shown in Figure 5(b). This file maintains information according to various protocols such as MSN, Facebook, and others. The current version of Digsby stores the SHA-1 hash as a 20-byte value in hexadecimal format, but previous versions use a mixture of hexadecimal and ASCII values along with special characters to store the SHA-1 hash. For example, *\x18\xc3\xec\x0f\xb2\x9d8\xa8O|\xce#\n\xc3\xef \xeb\xcax\xc3s* is an SHA-1 hash value in which *"\x"* indicates that the two following characters are hexadecimal values.

The information contained in the *iconhashes.dat* file is also linked with icons that are stored in the Digsby\Cache folder. This folder contains icons for each contact ID in a separate subfolder with a name inherited from the name of the underlying protocol such as Google Talk, MSN, and Yahoo, as shown in Table 5. These icons can be viewed using a Windows application like *Windows Photo Viewer*. Facebook icons are stored under the name *580xxx436_ICON.dat*, in which *580xxx436* represents the Facebook ID. Yahoo icons are kept under the name *alxxxxxkh_ICON.dat*, where *alxxxxxkh* is the Yahoo ID of the contact person *alxxxxxkh@yahoo.com*. Google Talk icons are stored with the complete address of the Gmail user, for instance, *yasxxxxxxns@gmail.com_ICON.dat*. Hotmail icon names such as *faxxxxxxab@hotmail.com_ICON.dat* are kept in the same format as Google Talk icons. The Facebook ID of a user can be found as a folder name at location *C:\Documents and Settings\<Windows_ID>\Local Settings\Digsby\cache\<Digsby ID>_ cache\fbchat\<protocol ID >*, which contains a *cookiejar* file preserving all cookies and their expiry time-stamps for

Table 2: Artifacts and their locations in Windows 7

Artifacts	Directory path
Default installation path, executables	C:\Program Files \Digsby C:\Users\<Windows_ID>\AppData\Local\Digsby\App <Directory Path>\Digsby\App
Cache, password, temporary data, icons, last usage time-stamp, Aliases	C:\Users\<Windows_ID>\AppData\Local\Digsby
File transfer	C:\Users\<Windows_ID>\Desktop
Emoticons, message styles	C:\Users\<Windows_ID>\AppData\Roaming\Digsby
Chat and update history	C:\Users\<Windows_ID>\Documents\Digsby Logs

Table 3: Artifacts and their directory paths for portable installed instances

Artifacts	Directory path
Default installation path, executables	<Directory path>\Digsby\App
Cache, proxy and configuration settings, temporary data, icons, last usage time-stamp, aliases, favorite icons	<Directory path>\Digsby\App\env \userlocaldata
File transfer	<Directory path>\Digsby\App\env \userdesktop
Emoticons, message styles	<Directory path>\Digsby\App\env \userdata
Chat and update history	<Directory path>\Digsby\App\env \Documents\Digsby Logs

the last login session.

3.2.4. Alias Names

Information about alternative names can be found in the *alias_cache_v1.db* file, which is stored at the *C:\Users\<Windows_ ID>\AppData\Local\Digsby\cache\<Digsby ID>_cache* location. This file can be opened using SQL LITE supporting software[8], as shown in Figure 6, to view the aliases of the contact IDs of various protocols. To protect users' privacy, some letters of the names and aliases have been replaced with the special character (?) in Figure 6.

3.2.5. E-mails and Social Activities

Digsby maintains only the artifacts of instant messages. It does not maintain records of sent and received e-mails and social activities performed over social networks. However, an examiner can collect the footprints of such activities from the Web browser history, as discussed in [22, 23], and related information can also be found in the Windows registry [18]. Process Monitor [9] can be used to monitor real-time activities performed on files, the registry, and processes to verify the results.

4. Digsby Password Decryption

This section discusses and provides implementation details of the password decryption techniques that can be used to decrypt encrypted passwords during the forensic analysis process. A discussion of possible insider attacks also constitutes an important part of this section.

4.1. Password Decryption for Machine-Specific Digsby

Digsby maintains login credentials in the *logininfo.yaml* file and does not permit users to change the default location of this file. This *"logininfo.yaml"* file contains the Digsby IDs of all users. It also contains the encrypted passwords of those who select the *save password* option. The current version of Digsby supports 31-character-long passwords, whereas previous versions supported only 16 characters. Digsby uses the RC4 encryption algorithm to encrypt a user's password and to generate base-64 code for the encrypted password before storing it into the designated file. The encryption and decryption key of the RC4 algorithm is a 20-byte SHA-1 hash of *system product ID*, *installation date*, and *Digsby ID*. The password cracking method presented in the article *Exposing the password secrets of Digsby* [24] can be summarized as follows:

1. Search for the *logininfo.yaml* file in the directory paths.
2. Decode the base-64 encoded passwords stored in the *logininfo.yaml* file.
3. Collect the system product IDs, installation dates, and Digsby IDs. The Digsby ID can be collected from the *logininfo.yaml* file, and the system product ID and installation date can be found in the Windows registry under the *HKEY_LOCAL_MACHINE \SOFTWARE\Microsoft\Windows NT\CurrentVersion* key using the *DigitalProductId* and *InstallDate* entries.
4. Create a complete string of system product ID, installation date, and Digsby ID in sequence as written and generate its SHA-1 hash, which serves as a decryption key for the encrypted password.
5. Decrypt the encrypted password using the RC4 encryption algorithm with the help of the generated SHA-1 hash key.

A work flow of the password cracking method discussed above is shown in Figure 7.

4.2. Password Decryption for Portable-Installed Digsby

Along the lines of the password decryption method for the machine-specific version of Digsby, the authors have devised a password decryption technique for the portable installation version of Digsby. During cryptanalysis of the algorithm used for encryption, an adaptive chosen-plaintext attack is launched. In an *adaptive chosen-plaintext attack*, the cryptanalyst can choose a series of plaintexts to be encrypted with an unknown algorithm and a key to acquire the respective cipher text. The

[8]http://portableapps.com/news/2010-12-01_-_sqlite_database_browser_portable_1.3

[9]http://technet.microsoft.com/en-us/sysinternals/bb896645

Table 4: Digsby directory paths for various instant messaging protocols

Platform	Directory path
Yahoo	C:\Users \<Windows_ID>\Documents\Digsby Logs\<Digsby ID>\yahoo\<Protocol ID>\<Contact ID>_yahoo
Google Talk	C:\Users \<Windows_ID>\Documents\Digsby Logs\<Digsby ID>\gtalk\<Protocol ID>@gmail.com\<Contact ID>@gmail.com_gtalk
Facebook	C:\Users \<Windows_ID>\Documents\Digsby Logs\<Digsby ID>\fbchat\<Protocol ID>@yahoo.com\69xxxx575_fbchat
WLM/MSN	C:\Users \<Windows_ID>\Documents\Digsby Logs\<Digsby ID>\msn\yaxxxxxns@hotmail.com\fahxxxx@hotmail.com_msn
Digsby	C:\Users \<Windows_ID>\Documents\Digsby Logs\<Digsby ID>\digsby\yaxxxxxns@digsby.org\digsby.org_digsby

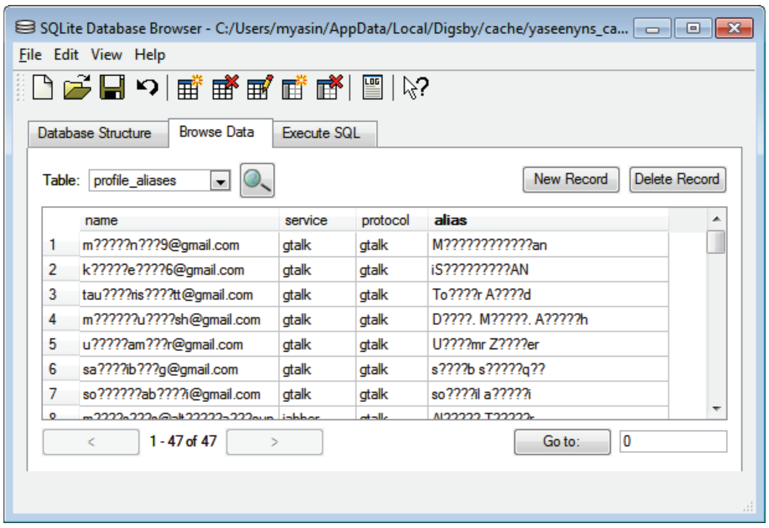


Table: profile_aliases

	name	service	protocol	alias
1	m?????n???9@gmail.com	gtalk	gtalk	M????????????an
2	k?????e????6@gmail.com	gtalk	gtalk	iS?????????AN
3	tau????ris????tt@gmail.com	gtalk	gtalk	To????r A????d
4	m??????u????sh@gmail.com	gtalk	gtalk	D????. M?????. A?????h
5	u?????am???r@gmail.com	gtalk	gtalk	U????mr Z????er
6	sa????ib???g@gmail.com	gtalk	gtalk	s????b s?????q??
7	so??????ab????i@gmail.com	gtalk	gtalk	so????il a?????i
8	[illegible]	jabber	gtalk	[illegible]

1 - 47 of 47

0

Figure 6: A snapshot showing aliases for contact IDs

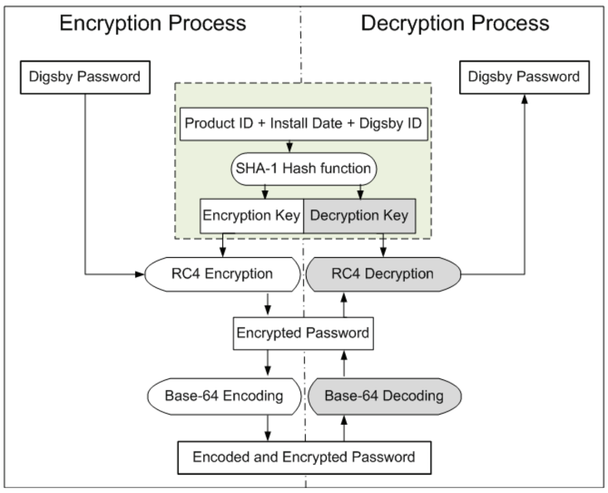


Figure 7: Password encryption and decryption process for machine-specific Digsby

analyst can choose successive plaintexts based on the results obtained from previous encryptions. During analysis of the Digsby encryption algorithm for the portable version, it was observed that this algorithm uses a symmetric encryption scheme, specifically XOR-encryption technique, to hide the login credentials of users with a fixed key. The mathematical function defined in Equation 1 states that the plaintext and key are XORed to generate the cipher text (encrypted password), where i represents the index value and l represents the total number of alphabets to be considered, or 31 in the case of Digsby. The *MOD* function is used to compute the remainder value after integer division, and the $\oplus$ function is used to compute the bitwise XOR of the current index value with the respective key value.

$$C[i] = P[i] \oplus K[i\ MOD\ l] \tag{1}$$

The symmetric nature of this encryption reveals that the same encryption key can be used to perform the decryption process, i.e., to extract the password in plaintext. As a result, Equation 2 can be used to decrypt the encrypted password, where the symbols have the same interpretations as in Equation 1.

$$P[i] = C[i] \oplus K[i\ MOD\ l] \tag{2}$$

Table 6 shows a few chosen plaintexts and the corresponding encrypted passwords for cryptanalysis of the cryptographic algorithm used to encrypt the passwords of the portable installation version of Digsby. The encrypted passwords are represented in the mixed format $\backslash x9b\backslash x12\backslash xdd\backslash x19\backslash xf8Qk\backslash x02\backslash x06\backslash xba$, which includes hexadecimal and ASCII values. In this string $\backslash x$ indicates that the next two characters are hexadecimal values and the remaining values are ASCII characters. These values must be converted into hexadecimal values, as shown in Table 7. The encrypted password at serial number 11 was used to confirm whether the left diagonal values from serial numbers 1 to 10 (shown in boldface) are similar to it. Note that the string consisting of the left diagonal values from serial numbers 1 to 10 is similar to the encrypted password given at serial number 11. For further clarification, another password at serial number 12 was chosen as a part of the chosen-plaintext attack to confirm the right diagonal values from serial numbers 1 to 10. The right diagonal values

Table 5: Icons-related artifacts for different protocols

Protocol	Folder name	Icons' name format	Protocol ID
Facebook	Fbchat	580xxx436_ICON.dat	580xxx436
Google	Gtalk	yasxxxxxxns@gmail.com.dat	yasxxxxxxns@gmail.com
Yahoo	Yahoo	alxxxxxkh_ICON.dat	alxxxxxkh
MSN	msn	faxxxxxxab@hotmail.com_ICON.dat	faxxxxxxab@hotmail.com

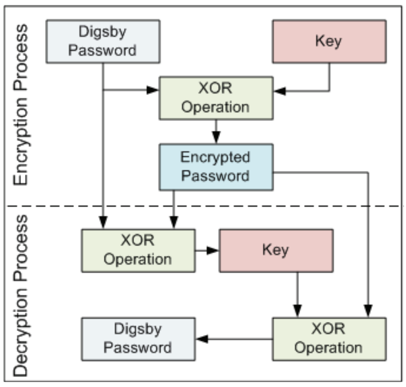


Figure 8: Password encryption and decryption process for Digsby portable installation version

also fully match with the launched attack. From this analysis, it can be concluded that Digsby possibly uses the XOR-encryption algorithm for encrypting passwords on a character-by-character basis, rather than on a block-by-block basis. Multiple passwords were chosen by different users to test and verify this encryption scheme for the portable installation version of Digsby.

Table 6: Chosen plaintexts and encrypted passwords

S. no.	Chosen plain-text	Encrypted password (Hex + ASCII format)
1	0000000000	**\x9b**\x12\xdd\x19\xf8Qk\x02\x06**\xba**
2	1111111111	\x9a**\x13**\xdc\x18\xf9Pj\x03**\x07**\xbb
3	2222222222	\x99\x10**\xdf**\x1b\xfaSi**\x00**\x04\xb8
4	3333333333	\x98\x11\xde**\x1a**\xfbR**h**\x01\x05\xb9
5	4444444444	\x9f\x16\xd9\x1d**\xfcU**o\x06\x02\xbe
6	5555555555	\x9e\x17\xd8\x1c**\xfdT**n\x07\x03\xbf
7	6666666666	\x9d\x14\xdb**\x1f**\xfeW**m**\x04\x00\xbc
8	7777777777	\x9c\x15**\xda**\x1e\xffVl**\x05**\x01\xbd
9	8888888888	\x93**\x1a**\xd5\x11\xf0Yc\n**\x0e**\xb2
10	9999999999	**\x92**\x1b\xd4\x10\xf1Xb\x0b\x0f**\xb3**
11	0123456789	\x9b\x13\xdf\x1a\xfcTm\x05\x0e\xb3
12	9876543210	\x92\x1a\xda\x1f\xfdUh\x00\x07\xba

After verification that the encryption algorithm performs only character-by-character replacement and does not have any impact on succeeding characters, the next step is to decode the encryption key. For the XOR-encryption algorithm, the decoding mechanism is very simple. The cryptanalyst needs only the chosen plaintext (password) and the encrypted password. Both the chosen plaintext and the encrypted password must be converted to pure hexadecimal format, as shown in Table 7. Then the XOR operation can be performed on both hexadecimal values to deduce the original key used by Digsby for encryption. The value of this key can be used to decrypt the passwords encrypted by the portable version of Digsby. Figure 8 presents a work flow of the password decryption process described above.

4.2.1. *Verification of the proposed key identification and password decryption algorithm*

To verify the accuracy of the proposed key identification and password decryption algorithm for the portable version of Digsby, a portable installation of Digsby was performed on different Windows machines, and ten different users were requested to create their own passwords and to use Digsby from their own dedicated computers. Users logged into Digsby using their credentials and performed various activities through Digsby. Afterwards, all passwords were collected from the *digsby.dat* file stored at the *\Digsby\App\env\userdata* location on the portable device. The decryption technique described above was used to decrypt their passwords, and the correctness of these passwords was confirmed by their respective users. In this experiment, all passwords were successfully decrypted using the proposed algorithm. A list of the encrypted and decrypted passwords along with their verification status is presented in Table 8.

4.3. *Insider Attack*

Digsby does not store the login credentials of instant messengers (IM), email accounts, and social networks on the user's computer, but rather it stores the hash values of the passwords on the server to avoid insider attacks, so that their own employees cannot view users' passwords. The login credentials of IM, email accounts, and social networks are encrypted using the plaintext of the password used to log in to Digsby. By analyzing Digsby artifacts at the client end, it could be observed that Digsby uses the SHA-1 hash function to generate hash values to decrypt stored passwords and contact icons, and it may also use the SHA-1 hash function to store password hash codes on the server. There are various methods of obtaining access to these login credentials. An insider with rights to view the hash passwords stored on the server can copy the password hash codes and put them on various websites [10] to generate passwords online. Once he/she obtains the password of a user in plaintext, there is an opportunity for an intruder to decrypt the login credentials of IMs, email addresses, and social networks if Digsby is using the RC4 encryption algorithm on the server side. Thereafter, the attacker can obtain the encrypted password from the database and thereby regenerate all saved passwords within a short period of time.

Figure 9 illustrates how an insider can launch an attack on the Digsby server. It is assumed that an insider has access to view the password hash codes stored in the Digsby database. In Figure 9, the symbols used have the following interpretations:

K_i = password of Digsby user i

[10]http://www.md5decrypter.co.uk/sha1-decrypt.aspx, http://hashcrack.com/

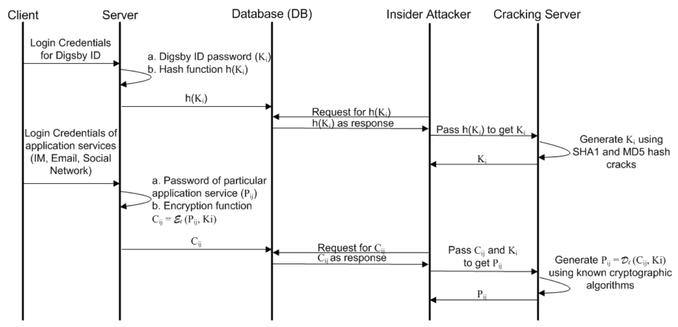

Client
Server
Database (DB)
Insider Attacker
Cracking Server
Login Credentials for Digsby ID
a. Digsby ID password (K_i)
b. Hash function $h(K_i)$
$h(K_i)$
Request for $h(K_i)$
$h(K_i)$ as response
Pass $h(K_i)$ to get K_i
Generate K_i using SHA1 and MD5 hash cracks
K_i
Login Credentials of application services (IM, Email, Social Network)
a. Password of particular application service (P_{ij})
b. Encryption function
$C_{ij} = \mathcal{E}_\ell (P_{ij}, Ki)$
C_{ij}
Request for C_{ij}
C_{ij} as response
Pass C_{ij} and K_i to get P_{ij}
Generate $P_{ij} = \mathcal{D}_\ell (C_{ij}, Ki)$ using known cryptographic algorithms
P_{ij}

Figure 9: Possible insider attacks

Table 7: Intermediate results during derivation of key from a chosen password

Item	**Values**
Encrypted password in Hexadecimal and ASCII formats	\x9b\x12\xdd\x19\xf8Qk\x02-\x06\xba\x85\xc5\xad\x9ec\xbb-T\x12\xe9\xa7\x85\xb0L\xea-J6\x06d\x7f\x1b\xf8
Encrypted Hexadecimal password	9B12DD19F8516B02-06BA85C5AD9E63BB-5412E9A785b04CEA-4A3606647F1BF8
Chosen password	00000000-00000000-00000000-0000000
Hexadecimal value of chosen password	3030303030303030-3030303030303030-3030303030303030-30303030303030
Key	AB22ED29C8615B32-368AB5F59DAE538B-6422D997B5807CDA-7A0636544F2BC8

$h(.)$ = a hash function, which can be either $h_{SHA1}(.)$ or $h_{MD5}(.)$
$h(K_i)$ = hash of the password of user i
$h^{-1}(K_i)$ = inverse hash of the password of user i
P_{ij} = password of user i for application j
$C_{ij} = E_\ell(P_{ij}, K_i)$= an encryption function, where $1 \leq \ell \leq P_{ij}$ represents an application-specific known encryption algorithm
$P_{ij} = D_\ell(C_{ij}, K_i)$ = a decryption function, where $1 \leq \ell \leq P_{ij}$ represents an application-specific known decryption algorithm.

Initially, a user registers on Digsby and obtains a unique user account, and account-related information, in particular login credentials, is stored on the Digsby server. The password component K_i of the user account is stored in a hash format $h(.)$ rather than in plaintext. Whenever a Digsby client wishes to log in to Digsby, he/she provides login credentials through a Web interface. The Digsby server applies the hash function $h(.)$ to the entered password K_i for login and sends a request to the related database to validate K_i and the associated Digsby ID. Because an insider may have access to the hashes of the user passwords, after obtaining access to the hash of a particular password $h(K_i)$, he/she can generate the password $h^{-1}(h(K_i))$, either through applying brute-force techniques or through exploiting the weaknesses of the hash functions used. Once it was discovered that Digsby uses the SHA-1 hash algorithm for various purposes, such as decryption of stored passwords and contact icons, on the client side, it could be presumed that Digsby might also use SHA-1 hash functions on the server side. If this is true, then an insider attacker can easily obtain the password hash $h(K_i)$ of a Digsby ID and can generate password K_i from this hash through exploiting the known hash function $h(.)$ and its weaknesses. Even if Digsby is not using the SHA-1 hash function on the server side or the hash function is robust enough to resist cracking, the insider can launch a brute-force attack to obtain the password $h^{-1}(h(Ki))$.

Moreover, Digsby saves the login credentials of supported IM applications, email accounts, and social networks on the Digsby server. These passwords are encrypted using the Digsby login password and stored in the Digsby database. Assuming that an insider has access to the Digsby database, in addition to obtaining access to the login password K_i of a user, he/she may send a query request to access the encrypted password C_{ij} for a particular application service. In this scenario, the attacker has both the login password K_i to log in to Digsby and the encrypted password C_{ij} to log in to a particular application service. Based on the known fact that Digsby uses the RC4 encryption algorithm to encrypt passwords stored on client machines, it may be assumed that the Digsby developers have used one of the known algorithms for encryption purposes on the client side, and consequently that they have used a similar or some other known algorithm for encryption purposes on the server side. Therefore, an inside attacker has only to confirm the known algorithm, and after that, in no time, he/she can obtain the login credentials, in particular the password C_{ij} of an application service.

5. RAM and Swap File Analysis

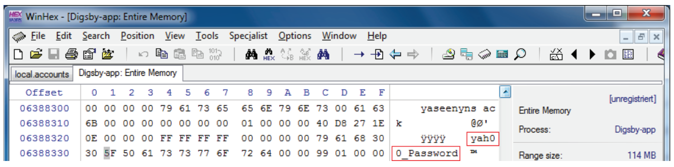
A large number of forensic artifacts can also be collected from RAM (random-access memory) and swap files during a live investigation process. RAM analysis can be valuable for finding login credentials, file paths, and proxy settings. Digsby stores essential information at various locations in RAM. During RAM analysis, it must be assumed that the data in memory change frequently according to users' activities. For example, it is possible that examiner finds a login credential for the MSN Messenger protocol at multiple positions during investigation in one case, whereas he/she may not be able to find even a single instance of login credentials in a second case. Such variations occur due to inconsistency in users' activities and the volatile nature of RAM. In this situation, it is necessary to conduct the analysis multiple times and to present the variations in results at the end. Another important fact related to RAM analysis is

Table 8: Accuracy of the password decryption process for Digsby portable installation version

User no.	Encrypted password	Decrypted password	Result
1	\xdbC\x9eZ\xbf\x0e)V\x01\xbd\x82\xc2	password7777	successful
2	\xfb\x11\x9f\x1e\x88\r:\\W\xe4\xdc\x9b\xf4\xfe2\xe5\x0eC\xb7 \xf0\xe6\xe5\x17\xbb\x16o	P3r7@lananiniPanjangSekali	successful
3	\xcaQ\x89O\xb9\x16>@B\xf3\x84\xc7\xae\x9af\xbd\x0eI\xb5\xac	asdfqwerty123456jkl;	successful
4	\xd9M\x98G\xac!/ZS\xbb\x87	round@the12	successful
5	\x9a\x13\xdd\x1a\x97Pb\n\x05	1103_1983	successful
6	\xfbb\xc9\r\xbfQ)V\x0e\xbb	P@$$w0rd81	successful
7	\xd8C\x80i\xf0Ti\x02	sam@8520	successful
8	\x90\x1a\x88G\xac,\x1bF_\xb1\xe9\xd7\xc6\xd3	;8endMti;"[}	successful
9	\xcaQ\x89O\xa2\n7\t\x07\xb8\x86\xc1\xa8\x98	asdfjkl;123456	successful
10	\xc6U\x85H\xa1\x089]v\xb9\x8d	mwhaiibo@38	successful

its volatile nature. An examiner may collect scattered information from different locations (e.g., the start, middle, and end) in RAM, and may use valuable search keywords to find specific information.

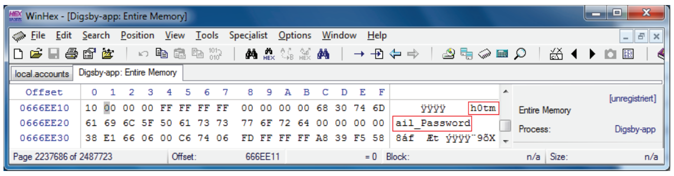
During analysis, it has been observed that an investigator can collect login credentials multiple times at different locations in RAM and swap files. These login credentials can be either encrypted or in plaintext. Initially, the examiner searches for unencrypted login credentials. If useful results are not found, then the search proceeds to encrypted passwords. The examiner can use special keywords to narrow down the search space and to collect the required artifacts. Figure 10 illustrates an occurrence of an unencrypted password for a Yahoo account to highlight the importance of RAM and swap file analysis during live examination. Similarly, Figure 11 shows the password for an MSN account in clear text. The highlighted *h0tmail_Password* in Figure 11 and *yah00_Password* in Figure 10 are the passwords in clear text. The examiner can use search keywords to open significant doorways for the collection of remaining artifacts, as shown in Table 9. The sample search keywords for the MSN account password are *login name*, *hotmail*, *login.live.com*, *live.com*, *msn*, and *<username>@hotmail.com*.


WinHex - [Digsby-app: Entire Memory]
File Edit Search Position View Tools Specialist Options Window Help
local.accounts
Digsby-app: Entire Memory

```
Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
06388300  00 00 00 00 79 61 73 65  65 6E 79 6E 73 00 61 63      yaseenyns ac
06388310  6B 00 00 00 00 00 00 00  01 00 00 00 40 D8 27 1E  k           @Ø'
06388320  0E 00 00 00 FF FF FF FF  00 00 00 00 79 61 68 30      ÿÿÿÿ    yah0
06388330  30 5F 50 61 73 73 77 6F  72 64 00 00 99 01 00 00  0_Password  ™
```

[unregistriert]
Entire Memory
Process: Digsby-app
Range size: 114 MB

Figure 10: Yahoo password in plaintext

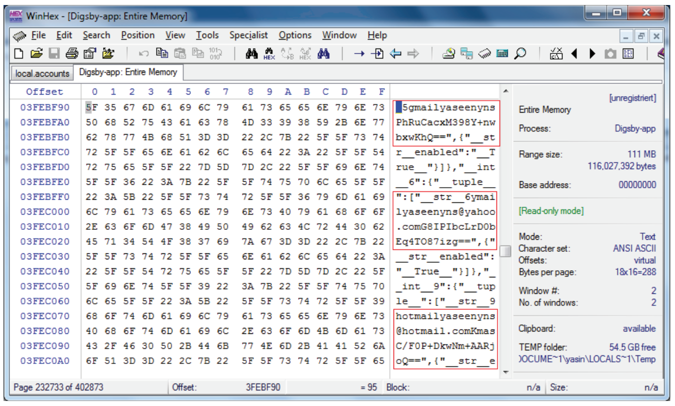

WinHex - [Digsby-app: Entire Memory]
File Edit Search Position View Tools Specialist Options Window Help
local.accounts
Digsby-app: Entire Memory

```
Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
0666EE10  10 00 00 00 FF FF FF FF  00 00 00 00 68 30 74 6D      ÿÿÿÿ    h0tm
0666EE20  61 69 6C 5F 50 61 73 73  77 6F 72 64 00 00 00 00  ail_Password
0666EE30  38 E1 66 06 00 C6 74 06  FD FF FF FF A8 39 F5 58  8áf  Æt ýÿÿÿ¨9õX
```

[unregistriert]
Entire Memory
Process: Digsby-app
Page 2237686 of 2487723 | Offset: 666EE11 | = 0 | Block: n/a | Size: n/a

Figure 11: Hotmail password in plaintext

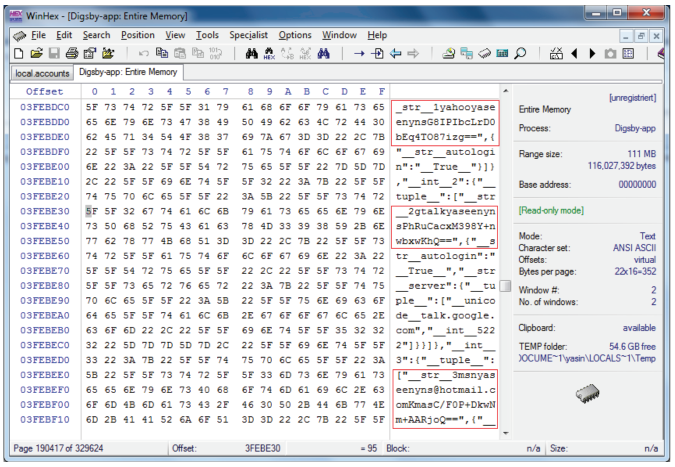
Figures 12 and 13 show occurrences of encrypted passwords for email and IM accounts in a case where the examiner does not find the required login credentials in plaintext. The investigator can search for keywords that contain the protocol name with the username, such as *<protocol Name><username>*. In Figure 12 *gmailyaseenyns*, *ymailyaseenyns*, and *hotmailyaseenyns* can be used as keywords to discover the encrypted passwords. Similarly, Figure 13 highlights the uses of the *yahooyaseenyns*, *gtalkyaseenyns*, and *msnyaseenyns* keywords to find encrypted passwords. Another keyword *_str_?* can also be concatenated with existing search keywords, where *?* is used to represent a wildcard character.


WinHex - [Digsby-app: Entire Memory]
File Edit Search Position View Tools Specialist Options Window Help
local.accounts
Digsby-app: Entire Memory

```
Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
03FEBF90  5F 35 67 6D 61 69 6C 79  61 73 65 65 6E 79 6E 73  _5gmailyaseenyns
03FEBFA0  50 68 52 75 43 61 63 78  4D 33 39 38 59 2B 6E 77  PhRuCacxM398Y+nw
03FEBFB0  62 78 77 4B 68 51 3D 3D  22 2C 7B 22 5F 5F 73 74  bxwKhQ==",{"__st
03FEBFC0  72 5F 5F 65 6E 61 62 6C  65 64 22 3A 22 5F 5F 54  r__enabled":"__T
03FEBFD0  72 75 65 5F 5F 22 7D 5D  7D 2C 22 5F 5F 69 6E 74  rue__"}]},"__int
03FEBFE0  5F 5F 36 22 3A 7B 22 5F  5F 74 75 70 6C 65 5F 5F  __6":{"__tuple__
03FEBFF0  22 3A 5B 22 5F 5F 73 74  72 5F 5F 36 79 6D 61 69  ":["__str__6ymai
03FEC000  6C 79 61 73 65 65 6E 79  6E 73 40 79 61 68 6F 6F  lyaseenyns@yahoo
03FEC010  2E 63 6F 6D 47 38 49 50  49 62 63 4C 72 44 30 62  .comG8IPIbcLrD0b
03FEC020  45 71 34 54 4F 38 37 69  7A 67 3D 3D 22 2C 7B 22  Eq4TO87izg==",{"
03FEC030  5F 5F 73 74 72 5F 5F 65  6E 61 62 6C 65 64 22 3A  __str__enabled":
03FEC040  22 5F 5F 54 72 75 65 5F  5F 22 7D 5D 7D 2C 22 5F  "__True__"}]},"_
03FEC050  5F 69 6E 74 5F 5F 39 22  3A 7B 22 5F 5F 74 75 70  _int__9":{"__tup
03FEC060  6C 65 5F 5F 22 3A 5B 22  5F 5F 73 74 72 5F 5F 39  le__":["__str__9
03FEC070  68 6F 74 6D 61 69 6C 79  61 73 65 65 6E 79 6E 73  hotmailyaseenyns
03FEC080  40 68 6F 74 6D 61 69 6C  2E 63 6F 6D 4B 6D 61 73  @hotmail.comKmas
03FEC090  43 2F 46 30 50 2B 44 6B  77 4E 6D 2B 41 41 52 6A  C/F0P+DkwNm+AARj
03FEC0A0  6F 51 3D 3D 22 2C 7B 22  5F 5F 73 74 72 5F 5F 65  oQ==",{"__str__e
```

[unregistriert]
Entire Memory
Process: Digsby-app
Range size: 111 MB 116,027,392 bytes
Base address: 00000000
[Read-only mode]
Mode: Text
Character set: ANSI ASCII
Offsets: virtual
Bytes per page: 18x16=288
Window #: 2
No. of windows: 2
Clipboard: available
TEMP folder: 54.5 GB free)OCUME~1\yasin\LOCALS~1\Temp
Page 232733 of 402873 | Offset: 3FEBF90 | = 95 | Block: n/a | Size: n/a

Figure 12: Encrypted passwords of email accounts


WinHex - [Digsby-app: Entire Memory]
File Edit Search Position View Tools Specialist Options Window Help
local.accounts
Digsby-app: Entire Memory

```
Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
03FEBDC0  5F 73 74 72 5F 5F 31 79  61 68 6F 6F 79 61 73 65  _str__1yahooyase
03FEBDD0  65 6E 79 6E 73 47 38 49  50 49 62 63 4C 72 44 30  enynsG8IPIbcLrD0
03FEBDE0  62 45 71 34 54 4F 38 37  69 7A 67 3D 3D 22 2C 7B  bEq4TO87izg==",{
03FEBDF0  22 5F 5F 73 74 72 5F 5F  61 75 74 6F 6C 6F 67 69  "__str__autologi
03FEBE00  6E 22 3A 22 5F 5F 54 72  75 65 5F 5F 22 7D 5D 7D  n":"__True__"}]}
03FEBE10  2C 22 5F 5F 69 6E 74 5F  5F 32 22 3A 7B 22 5F 5F  ,"__int__2":{"__
03FEBE20  74 75 70 6C 65 5F 5F 22  3A 5B 22 5F 5F 73 74 72  tuple__":["__str
03FEBE30  5F 5F 32 67 74 61 6C 6B  79 61 73 65 65 6E 79 6E  __2gtalkyaseenyn
03FEBE40  73 50 68 52 75 43 61 63  78 4D 33 39 38 59 2B 6E  sPhRuCacxM398Y+n
03FEBE50  77 62 78 77 4B 68 51 3D  3D 22 2C 7B 22 5F 5F 73  wbxwKhQ==",{"__s
03FEBE60  74 72 5F 5F 61 75 74 6F  6C 6F 67 69 6E 22 3A 22  tr__autologin":"
03FEBE70  5F 5F 54 72 75 65 5F 5F  22 2C 22 5F 5F 73 74 72  __True__","__str
03FEBE80  5F 5F 73 65 72 76 65 72  22 3A 7B 22 5F 5F 74 75  __server":{"__tu
03FEBE90  70 6C 65 5F 5F 22 3A 5B  22 5F 5F 75 6E 69 63 6F  ple__":["__unico
03FEBEA0  64 65 5F 5F 74 61 6C 6B  2E 67 6F 6F 67 6C 65 2E  de__talk.google.
03FEBEB0  63 6F 6D 22 2C 22 5F 5F  69 6E 74 5F 5F 35 32 32  com","__int__522
03FEBEC0  32 22 5D 7D 7D 5D 7D 2C  22 5F 5F 69 6E 74 5F 5F  2"]}}]},"__int__
03FEBED0  33 22 3A 7B 22 5F 5F 74  75 70 6C 65 5F 5F 22 3A  3":{"__tuple__":
03FEBEE0  5B 22 5F 5F 73 74 72 5F  5F 33 6D 73 6E 79 61 73  ["__str__3msnyas
03FEBEF0  65 65 6E 79 6E 73 40 68  6F 74 6D 61 69 6C 2E 63  eenyns@hotmail.c
03FEBF00  6F 6D 4B 6D 61 73 43 2F  46 30 50 2B 44 6B 77 4E  omKmasC/F0P+DkwN
03FEBF10  6D 2B 41 41 52 6A 6F 51  3D 3D 22 2C 7B 22 5F 5F  m+AARjoQ==",{"__
```

[unregistriert]
Entire Memory
Process: Digsby-app
Range size: 111 MB 116,027,392 bytes
Base address: 00000000
[Read-only mode]
Mode: Text
Character set: ANSI ASCII
Offsets: virtual
Bytes per page: 22x16=352
Window #: 2
No. of windows: 2
Clipboard: available
TEMP folder: 54.6 GB free)OCUME~1\yasin\LOCALS~1\Temp
Page 190417 of 329624 | Offset: 3FEBE30 | = 95 | Block: n/a | Size: n/a

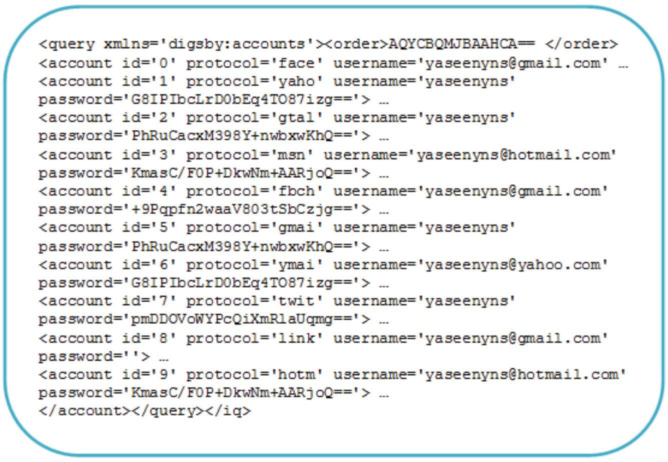
Figure 13: Encrypted passwords of IM accounts

The examiner can also search for essential directory paths as supporting information which has its own importance during RAM and swap file analyses. The email addresses of current Digsby users can also be found during analysis to help the examiner to trace the association of suspect persons with other

Table 9: Search keywords and the number of instances of login credentials

Protocol	Login credentials	Search keywords
Yahoo	1 – 4 times	Email address
Google talk	1 time	Msn, @hotmail
Hotmail	4 – 6 times	Gmail, <gmail>, (gmail)
Facebook	2 – 3 times	(facebook), (fbchat)

people in his/her network. Moreover, an investigator can also collect notification messages about new emails for configured email accounts. These notification messages can be searched for using the keywords *<email address>@<protocol>* and *new messages*. Note that an examiner cannot collect login credentials unless an instance of Digsby is running and a particular user is logged in. The information provided in Table 9 summarizes the results gathered during analysis. The frequency count values can vary with respect to the status of RAM updates and user activities. During analysis, encrypted login credentials of the user on all IM accounts, emails, and social networks were also found in a single location in RAM, as shown in Figure 14. *FR_PRIVATE* can be used to discover the passwords for the underlying applications.



```
<query xmlns='digsby:accounts'><order>AQYCBQMJBAAHCA== </order>
<account id='0' protocol='face' username='yaseenyns@gmail.com' ...
<account id='1' protocol='yaho' username='yaseenyns'
password='G8IPIbcLrD0bEq4TO87izg=='> ...
<account id='2' protocol='gtal' username='yaseenyns'
password='PhRuCacxM398Y+nwbxwKhQ=='> ...
<account id='3' protocol='msn' username='yaseenyns@hotmail.com'
password='KmasC/F0P+DkwNm+AARjoQ=='> ...
<account id='4' protocol='fbch' username='yaseenyns@gmail.com'
password='+9Pqpfn2waaV803tSbCzjg=='> ...
<account id='5' protocol='gmai' username='yaseenyns'
password='PhRuCacxM398Y+nwbxwKhQ=='> ...
<account id='6' protocol='ymai' username='yaseenyns@yahoo.com'
password='G8IPIbcLrD0bEq4TO87izg=='> ...
<account id='7' protocol='twit' username='yaseenyns'
password='pmDDOVoWYPcQiXmRlaUqmg=='> ...
<account id='8' protocol='link' username='yaseenyns@gmail.com'
password=''> ...
<account id='9' protocol='hotm' username='yaseenyns@hotmail.com'
password='KmasC/F0P+DkwNm+AARjoQ=='> ...
</account></query></iq>
```

Figure 14: A snapshot of RAM content showing login credentials of email, IM, and social network accounts

6. Conclusions

This paper has presented the design of a forensic tool, `DigLA`, which can assist forensic examiners to analyze Digsby log data efficiently through a user-friendly graphical user interface. `DigLA` assists in Windows registry analysis for search pages, toolbars, uninstall paths and information about recent Digsby usage. Examiners can also view Digsby log files, including installation paths, executable files, login credentials, proxy and configuration settings, last usage time-stamps, aliases of contact IDs, exchanged files, chat logs, and contact icons. `DigLA` is further enriched with a password-decryption process for both machine-specific and portable Digsby installation versions. During the analysis and artifact collection process, it was noted that all artifacts were not located on a single Windows directory path. Therefore, `DigLA` could provide a helping hand for forensic examiners to access each and every pertinent directory without providing any opportunity for evidence loss. A detailed discussion of an expected insider attack was also presented. Analyses of RAM and swap files were performed to collect user login credentials for Digsby and underlying applications.

[1] Carvey, H.. Instant messaging investigations on a live windows xp system. Digital Investigation 2004;1(4):256–260.

[2] Belkasoft, . Forensic instant messenger investigation. 2009. URL `http://www.belkasoft.com`.

[3] Dickson, M.. An examination into yahoo messenger 7.0 contact identification. Digital Investigation 2006;3(3):159–165.

[4] Levendoski, M., Datar, T., Rogers, M., Huff, D.. Yahoo messenger forensics 2011;.

[5] Dickson, M.. An examination into msn messenger 7.5 contact identification. Digital Investigation 2006;3(2):79–83.

[6] Van Dongen, W.. Forensic artefacts left by windows live messenger 8.0. Digital Investigation 2007;4(2):73–87.

[7] Parsonage, H.. The forensic recovery of instant messages from msn messenger and windows live messenger. 2008. URL `http://computerforensics.parsonage.co.uk/default.html`.

[8] Dickson, M.. An examination into aol instant messenger 5.5 contact identification. Digital Investigation 2006;3(4):227–237.

[9] Reust, J.. Case study: Aol instant messenger trace evidence. Digital Investigation 2006;3(4):238–243.

[10] Gao, Y., Cao, T.. Memory forensics for qq from a live system. Journal of Computers 2010;5(4):541–548.

[11] Yang, Y., Chow, K., Hui, L., Wang, C., Chen, L., Chen, Z., et al. Forensic analysis of popular chinese internet applications. Advances in Digital Forensics VI 2010;:285–295.

[12] Baset, S., Schulzrinne, H.. An analysis of the skype peer-to-peer internet telephony protocol. Arxiv preprint cs/0412017 2004;.

[13] Van Dongen, W.. Forensic artefacts left by pidgin messenger 2.0. Digital Investigation 2007;4(3):138–145.

[14] Dickson, M.. An examination into trillian basic 3. x contact identification. Digital Investigation 2007;4(1):36–45.

[15] Kiley, M., Dankner, S., Rogers, M.. Forensic analysis of volatile instant messaging. Advances in Digital Forensics IV 2008;:129–138.

[16] Husain, M., Sridhar, R.. iforensics: Forensic analysis of instant messaging on smart phones. Digital Forensics and Cyber Crime 2010;:9–18.

[17] Farmer, D.. A forensic analysis of the windows registry. 2007. [Last Accessed On: 03/03/2012]; URL `http://eptuners.com/forensics/contents/examination.htm`.

[18] AccessData, . Registry quick find chart. 2010. URL `http://accessdata.com/`.

[19] Yasin, M., Wahla, M., Kausar, F.. Analysis of download accelerator plus (dap) for forensic artefacts. In: Proceedings of the 5th International Conference on IT Security Incident Management and IT Forensics (IMF'09). IEEE; 2009, p. 142–152.

[20] Yasin, M., Wahla, M., Kausar, F.. Analysis of free download manager for forensic artefacts. Digital Forensics and Cyber Crime 2010;:59–68.

[21] Yasin, M., Cheema, A., Kausar, F.. Analysis of internet download manager for collection of digital forensic artefacts. Digital Investigation 2010;7(1):90–94.

[22] Pyne, S.. Internet explorer forensics: Reconstructing internet activity using pasco and galleta. 2007.

[23] Oh, J., Lee, S., Lee, S.. Advanced evidence collection and analysis of web browser activity. Digital Investigation 2011;8:S62–S70.

[24] SecurityXploded, . Exposing the password secrets of digsby. 2010. URL `http://securityxploded.com/digsbypasswordsecrets.php`.