Санкт-Петербургский государственный университет

Программная инженерия

Гогина Олеся Юрьевна

Анализ данных Snapchat на iOS

Курсовая работа

Научный руководитель:
доц., к.т.н. Литвинов Ю. В.

Консультант:
рук. отд. раз. ПО, ООО “Белкасофт” Тимофеев Н. М.

Санкт-Петербург
2020

Оглавление

Введение

В современном мире общение в основном происходит в социальных сетях. Наличие смартфона почти у каждого человека позволяет людям оставаться на связи, находясь в разных точках планеты. К сожалению, преступники также стали использовать современные технологии для общения с сообщниками и планирования преступлений и правонарушений. Особый интерес для криминального мира представляют средства связи, позволяющие скрыть содержание диалога пользователей. Одно из таких средств — Snapchat [10], сервис обмена моментальными сообщениями, ключевой функцией которого является самоуничтожение отправленных/полученных сообщений. Главной уязвимостью таких средств связи является физическое хранение части сообщений непосредственно на устройствах пользователей, так как в случае изъятия смартфона с установленным на него мессенджером, криминалисты имеют возможность получить полную копию файловой системы устройства. Данная копия содержит в себе всю информацию, хранимую на устройстве, в том числе данные мессенджеров.

Для хранения данных разработчики современных мессенджеров создают собственные, подходящие для конкретного приложения форматы, варьирующиеся в зависимости от операционной системы устройства. Так как данные форматы хранения данных являются частью проприетарного программного обеспечения, они не публикуются в открытом доступе. Одной из задач цифровой криминалистики является получение сведений о таких форматах и реализация инструментов для извлечения данных соответствующего приложения.

С целью развития приложений и повышения их защищенности от неправомерного доступа разработчики регулярно выпускают обновления для форматов хранения данных, используемых в их приложениях. В одном из таких обновлений разработчики Snapchat ввели в эксплуатацию новый формат хранения данных с заголовком TSAF [6]. Данный формат используется для хранения информации о диалогах пользователей приложения Snapchat на устройствах под управлением операци-

онной системы iOS. Данные об участниках диалогов и содержимое всех сообщений сохраняются в файл chatConversationStore.plist.

Существующие решения для извлечения данных Snapchat, сохраненных в формате TSAF, из полного логического образа файловой системы устройства под управлением операционной системы iOS, такие как Spoopy [8] и Cellebrite UFED Physical Analyzer [11] позволяют получить доступ только к текстовым сообщениям и не способны работать с видео, изображениями, геолокациями и другими не текстовыми данными, отправленными или полученными пользователями Snapchat.

В данной работе будет предложено комплексное решение для извлечения всех данных Snapchat из файла формата TSAF, полученного при помощи снятия полного логического образа файловой системы смартфона под управлением операционной системы iOS.

1. Постановка задачи

Цель работы — изучить проприетарный формат хранения данных TSAF и реализовать прототип программы для извлечения данных из записей формата TSAF.

Для достижения цели были поставлены следующие задачи:

1. исследовать, а затем сделать обзор нового формата данных TSAF;
2. реализовать прототип парсера файла TSAF;
3. разработать прототип для извлечения сообщений приложения Snapchat из снимка памяти iOS;
4. апробировать реализованный прототип на тестовом смартфоне iPhone 5S.

2. Обзор

В данной главе будет рассмотрен стандартный формат хранения данных приложения Apple binary plist [2], существующие на данный момент решения по извлечению и анализу данных приложения Snapchat на iOS и способ извлечения всех файлов данного приложенгия из памяти смартфона.

2.1. Формат Plist

Формат разработан компанией Apple. Название расширения Plist происходит от английского «Property List», на русский язык переводится как список свойств. Этот формат широко используется приложениями и другим программным обеспечением на OS X и iOS. Элементы списка могут быть представлены в виде примитивных значений (строки, числа, двоичные данные, даты и логические значения), либо в виде контейнеров для элементов (массивы, ассоциативные массивы). Контейнеры могут содержать другие контейнеры, а также примитивные типы. Корневой объект списка свойств находится наверху этой иерархии, и почти во всех случаях это массив или ассоциативный массив [1].

Список свойств может храниться одним из трех разных способов: в XML-представлении, в двоичном формате или в новом формате TSAF.

2.1.1. XML Plist

Файлы Plist, сохраненные в XML-представлении, удобны для чтения, имеют стандартный формат XML и поддерживаются встроенным редактором списка свойств Xcode. Структура XML Plists начинается со стандартной информации заголовка и содержит один корневой элемент, заключенный в тег типа документа <plist>. Элемент <plist> также содержит ровно один объект, обозначенный одним из элементов XML. При кодировании словарей элемент <key> используется для ключей словаря, а один из других тегов списка свойств используется для значения ключа [3].

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist SYSTEM "file://localhost/System/Library/DTDs/PropertyList.dtd">
<plist version="1.0">
    <dict>
        <key>Author</key>
        <string>William Shakespeare</string>
        <key>Lines</key>
        <array>
            <string>It is a tale told by an idiot,</string>
            <string>Full of sound and fury, signifying nothing.</string>
        </array>
        <key>Birthdate</key>
        <integer>1564</integer>
    </dict>
</plist>
```

Рис. 1: Пример сгенерированного XML Plist из списка свойств

2.1.2. Binary Plist

Двоичные файлы .plist занимают меньше места на диске, поэтому операционная система Mac OS X получает к ним доступ и использует их быстрее, чем их XML-аналоги. В связи с высокими требованиями к производительности операционной системы Mac OS X двоичные файлы .plist являются востребованными в сборках данной операционной системы. Заголовок файла (первые несколько байт в определенной последовательности) состоит из 8 байт, содержащих «bplist» и версию данного формата. Таблица объектов содержит все объекты данного формата. Все типы объектов определяются маркером (см. рис. 2). Маркер кодирует информацию о типе объекта и размере типа объекта. OFFSET TABLE содержит смещения к таблице объектов, тем самым приводит к фактическим значениям объектов. TRAILER имеет длину 32 байта и содержит информацию о размере.

2.1.3. TSAF Plist

Файл формата TSAF имеет расширение plist, но не отформатирован как binary plist или XML plist. Заголовок файла состоит из 4 байт, давших название новому формату: «TSAF». Способ организации данных сильно отличается от представлений списка свойств, рассмотренных выше.

```
HEADER
        magic number ("bplist")
        file format version

OBJECT TABLE
        variable-sized objects

        Object Formats (marker byte followed by additional info in some cases)
        null    0000 0000
        bool    0000 1000                       // false
        bool    0000 1001                       // true
        fill    0000 1111                       // fill byte
        int     0001 nnnn       ...             // # of bytes is 2^nnnn, big-endian bytes
        real    0010 nnnn       ...             // # of bytes is 2^nnnn, big-endian bytes
        date    0011 0011       ...             // 8 byte float follows, big-endian bytes
        data    0100 nnnn       [int]   ...     // nnnn is number of bytes unless 1111 then int count follows, followed by bytes
        string  0101 nnnn       [int]   ...     // ASCII string, nnnn is # of chars, else 1111 then int count, then bytes
        string  0110 nnnn       [int]   ...     // Unicode string, nnnn is # of chars, else 1111 then int count, then big-endian 2-byte uint16_t
                0111 xxxx                       // unused
        uid     1000 nnnn       ...             // nnnn+1 is # of bytes
                1001 xxxx                       // unused
        array   1010 nnnn       [int]   objref* // nnnn is count, unless '1111', then int count follows
                1011 xxxx                       // unused
        set     1100 nnnn       [int]   objref* // nnnn is count, unless '1111', then int count follows
        dict    1101 nnnn       [int]   keyref* objref* // nnnn is count, unless '1111', then int count follows
                1110 xxxx                       // unused
                1111 xxxx                       // unused

OFFSET TABLE
        list of ints, byte size of which is given in trailer
        -- these are the byte offsets into the file
        -- number of these is in the trailer

TRAILER
        byte size of offset ints in offset table
        byte size of object refs in arrays and dicts
        number of offsets in offset table (also is number of objects)
        element # in offset table which is top level object
        offset table offset
```

Рис. 2: Двоичное представление поддерживаемых маркеров bplist (из CFBinaryPList.c) [2]

2.2. Существующие инструменты, реализующие извлечение и анализ данных приложения Snapchat

Одна из первых работ в области извлечения данных приложения Snapchat — проприетарный парсер от Cellebrite [11]. Краткий обзор формата TSAF сделал Ian Whiffin [6], авторитетный специалист в области компьютерной криминалистики [12]. По результатам данного обзора был реализован парсер текстовых сообщений Spoopy [8]. Однако все рассмотренные инструменты позволяют получить доступ только к текстовым сообщениям и не способны работать с видео, изображениями, геолокациями и другими не текстовыми данными, отправленными или полученными пользователями Snapchat.

2.2.1. Spoopy

Spoopy — инструмент, находящийся в разработке и предназначенный для извлечения данных из приложения Snapchat в удобный для использования формат. Извлеченные данные о диалогах пользователей

помещаются в табличное представление с возможностью поиска по дате, фильтрацией результатов, автоматического перевода содержимого сообщения с помощью Google Translate.

2.2.2. Cellebrite UFED Physical Analyzer

Cellebrite UFED Physical Analyzer — проприетарный инструмент от компании Cellebrite. Данный инструмент позволяет извлекать данные приложения Snapchat и представлять их в виде древовидной структуры. Представление диалогов в таком виде является сложным для восприятия, так как древовидная структура не позволяет рассматривать сообщения пользователей как цельный диалог.

2.3. Извлечение файлов приложения Snapchat

Устройства, с установленной на них операционной системой iOS, отличаются особой защищенностью и закрытостью. На таких устройствах отсутствует прямой доступ к файловой системе [13]. Чтобы извлечь файлы приложения Snapchat, необходимо установить специальное приложение-агент Belkasoft Evidence Center [5]. Затем подключить телефон к компьютеру, выбрать в данном инструменте опцию — снятие образа с помощью агента. Полученный образ данных в формате tar содержит папку с файлами приложения Snapchat.

3. Формат TSAF

Файл рассматриваемого формата состоит из заголовка «TSAF» и данных приложения. Данные представляют собой снимок объекта, используемого для хранения данных о диалогах пользователя.

Объект содержит поля различных типов, часть из которых также являются объектами, реализующими специальные, определяемые в данном файле классы.

Для формирования снимка в файл сохраняются значения всех полей, а для объектов, реализующих определенные в файле формата TSAF классы, дополнительно сохраняется описание класса.

В этом разделе будет описан процесс исследования данного формата и типы данных, которые используются в файлах формата TSAF.

3.1. Исследование формата TSAF

В качестве основы для исследования даного формата была использована статья "SnapChat PList" [7], в которой автор описывает основную структуру содержимого файла формата TSAF для приложения Snapchat и приводит свои идеи, связанные с парсингом данного формата. Однако данная работа дает лишь общее представление об организации данных в формате TSAF.

Для более подробного изучения структуры хранения данных в формате TSAF на основе статьи Ian Whiffin был реализован базовый парсер данного формата, однако в ходе реализации были выявлены типы, не описанные в статье. Для классификации данных типов использовалось выравнивание полей, соответствующих неизвестному типу, в файле. Впоследствии точный тип таких полей устанавливался за счет определения информации, которую приложение Snapchat хранит в данных полях.

3.2. Используемые в формате TSAF типы данных

Объект, снимок которого сохраняется в файле формата TSAF, может содержать поля различных типов. Тип данных каждого поля объекта объявляется в определении класса, который реализует объект, либо непосредственно перед значением поля с помощью специальной сигнатуры.

Полями объектов могут являться примитивные типы данных, указатели, массивы, словари и другие объекты. Также поле объекта может не содержать никаких данных (NULL), в таком случае в него записывается байт «0x00».

3.2.1. Примитивные типы данных

В файле формата TSAF могут встречаться следующие примитивные типы данных:

- логический тип данных — 1 байт, содержащий значение «0x0D» (истина) или «0x0E» (ложь);
- строка, ее начало обозначается сигнатурой «0x08», а конец — байтом «0x00»;
- целочисленные типы данных (записываются в порядке Little Endian):
  - Int8 — 1 байт, его начало может обозначаться сигнатурой «0x0F»,
  - Int16 — 2 байта, его начало может обозначаться сигнатурой «0x10»,
  - Int32 — 4 байта, его начало может обозначаться сигнатурой «0x04»,
  - Int64 — 8 байт, его начало может обозначаться сигнатурой «0x12»;
- числа с плавающей запятой (записываются в порядке Little Endian):

– Single — 4 байта, его начало может обозначаться сигнатурой «0x13»,

– Double — 8 байт, его начало может обозначаться сигнатурой «0x14» или «0x16»,

Все численные типы данных имеют выравнивание, равное занимаемому ими количеству байт.

3.2.2. Указатели

Для экономии памяти, занимаемой файлом в формате TSAF, применяются указатели на уже упомянутые значения и строки.

Указатели делятся на две группы: указатели на строки и указатели на значения.

Указатели на строки объявляются с помощью сигнатуры «0x05» (значение такого указателя представлено как UInt8) и с помощью сигнатуры «0x06» (значение такого указателя представлено как UInt16). Указатель содержит номер строки среди всех строк, встреченных с начала файла, нумерация начинается с нуля.

Указатели на значения могут указывать на Single, Double (объявленный с сигнатурой «0x14»), Int64, словарь и массив. Данные указатели объявляются с помощью сигнатуры «0x02» (значение такого указателя представлено как UInt8) и с помощью сигнатуры «0x03» (значение такого указателя представлено как UInt16).

Указатель содержит номер значения среди всех значений типов, на которые может ссылаться данный указатель, встреченных с начала файла, нумерация начинается с нуля.

3.2.3. Массивы и словари

Для хранения набора значений в файлах формата TSAF используются массивы и словари. Массив объявляется с помощью сигнатуры «0x0A», словарь помечается сигнатурой «0x09». За сигнатурой следует целое число (Int32), определяющее количество элементов в массиве

или словаре. Элементы массива (пары «значение»:«ключ» для словарей) располагаются последовательно друг за другом сразу после данного числа.

Для представления наборов элементов, длина которых заранее неизвестна, используется специальная структура — именованный словарь. Данная структура обозначается сигнатурой «0x2C» и является парой «имя»:«словарь», где «имя» — строка (или указатель на строку), являющаяся названием структуры, а словарь — набор пар «ключ»:«значение» заранее неизвестной длины, конец которого обозначается сигнатурой «0x00».

Для сохранения информации в необработанном виде в файле формата TSAF используется специальный массив байт. Его началом служит сигнатура «0x15» после которой следует количество байт в данном массиве (Int32), а затем последовательно его элементы.

3.2.4. Классы

Объявление класса происходит при первом упоминании объекта, который реализует данный класс. Тело объявления помещается непосредственно перед определением объекта и помечается специальной сигнатурой «0x1E» в начале. После сигнатуры класса размещается строка, являющаяся названием объявляемого класса. В теле класса сразу после названия расположено число, имеющее тип Int32, обозначающее количество полей в классе, первый бит которого всегда равен единице. Далее следует определение типов полей класса, разделенное на 8-байтовые сегменты. Первый байт каждого сегмента описывает ожидаемый тип данных в соответствующем поле класса.

Возможные типы данных, объявленные в определении класса:

- байт «0x00» обозначает, что поле имеет динамический тип данных (он будет объявлен с помощью сигнатуры перед началом значения поля в объекте данного класса),
- байт «0x01» обозначает, что поле имеет логический тип данных и

использует 1 байт,

- байт «0x04» обозначает, что поле имеет целочисленный тип данных (Int32) и использует 4 байта,
- байт «0x05» обозначает, что поле представляет собой число с плавающей запятой с двойной точностью (Double) и использует 8 байт,
- байт «0x09» обозначает, что поле имеет целочисленный тип данных (Int64) и использует 8 байта.

3.2.5. Объекты

Объекты используются для представления информации в формате, используемом в приложении Snapchat. Каждый объект реализует некоторый предварительно объявленный в файле класс. Сигнатурой начала объявления объекта является «0x1F». После сигнатуры следует порядковый номер класса, который реализует объект. Номера получают все классы, упоминания которых встречались до данного объявления объекта, нумерация ведется с нуля. После ссылки на класс следуют объявления всех полей объекта.

4. Использование формата TSAF приложением Snapchat

Основной интерес при исследовании представляют данные, которые приложение Snapchat хранит в файлах формата TSAF. Таким образом, помимо определения типов полей и общей структуры файла формата TSAF, в ходе работы необходимо было провести соответствие между данными приложения и результатом парсинга данного файла. В работе было проведено исследование файла chatConversationStore.plist с целью извлечения информации о диалогах пользователя. В данном разделе описано как проводилось исследование и сопоставление результатов парсинга файла chatConversationStore.plist и информации о диалогах пользователя приложения Snapchat, а также приведено описание структуры файла, хранящего информацию о диалогах пользователя в формате TSAF.

4.1. Исследование данных приложения Snapchat

Для анализа данных приложения Snapchat оно было установлено на тестовый смартфон iPhone 5S, предоставленный компанией Belkasoft[4].

В данном приложении были созданы тестовые диалоги с другими пользователями. Также при изучении результатов парсинга файла chatConversationStore.plist были определены объекты, хранящие информацию о диалогах и конкретных сообщениях.

Это было возможно за счет именования классов, которые эти объекты реализуют (SCChatConversationV3 и SCChatMessageV3).

Сопоставляя данные диалога (например имя собеседника, время отправки сообщения, текст сообщения) и данные, полученные в ходе парсинга файла chatConversationStore.plist, удалось определить поля, которые хранят информацию об:

- имени собеседника;
- тексте сообщения;

- времени отправки сообщения;
- приложенной геолокации;
- приложенных медиа файлах;
- приложенных эмоджи, стикерах и битмоджи.

Несмотря на то, что файл chatConversationStore.plist содержит информацию о пересылаемых медиафайлах, сами файлы располагаются отдельно в папке приложения Snapchat. Для сопоставления медиафайлов и сообщений, в которых они были отправлены, используется встроенная база данных SQLite [9]. При этом сами медиафайлы хранятся без расширения в имени файла, что затрудняет их поиск в папке приложения Snapchat, однако, зная содержимое изображения из тестового диалога, в ходе работы удалось обнаружить файл, содержащий это изображение, в папке приложения, а затем сопоставить хранимые в информации о сообщении данные и имя найденного файла с изображением.

4.2. Структура файла для приложения Snapchat

Объект, снимок которого расположен в файле, является именованным словарем. В контексте данной работы стоит отметить две ключевые пары «ключ»:«значение», содержащиеся в данном словаре:
«conversations»:«словарь диалогов» и «username»:«имя пользователя», где «имя пользователя» — имя владельца аккаунта приложения Snapchat, а «словарь диалогов» — набор пар «ключ»:«значение», сопоставляющих имя собеседника и объект, хранящий информацию о диалоге с ним.

Одним из полей объекта, хранящего информацию о диалоге, является массив сообщений — объектов, содержащих информацию об одном сообщении из данного диалога. Для определения точного номера поля в объекте, необходимо проверить все поля, являющиеся массивами. При такой проверке необходимо определить тип элементов массива: если

	Тип данных	**Содержимое**
1	Объект класса «SCChatSticker»	Информация о стикере, приложенном к сообщению
2	Объект класса «SCChatMediaContent»	Информация о медиафайле, приложенном к сообщению
3	Объект класса «SCChatMessageParcel»	Геолокация, приложенная к сообщению
4	Строка	Имя отправителя
5	Число с плавающей запятой двойной точности	Время отправки, клентское
6	Словарь, сопоставляющий имена пользователей и объекты класса «SCChatMessageSavedState»	Кем было сохранено сообщение
7	Число с плавающей запятой двойной точности	Время отправки, серверное
8	Строка	Текст сообщения

Таблица 1: Интересующие специалистов кибербезопасности поля объекта, хранящего данные о сообщении

элемент является объектом класса «SCChatMessageV3», то это искомый массив сообщений. Если в ходе проверки не было обнаружено ни одного массива, удовлетворяющего условию, то это означает отсутствие загруженных на смартфон, из памяти которого был извлечен исследуемый файл, сообщений в диалоге с данным пользователем.

Каждое сообщение в диалоге представлено объектом, реализующим класс «SCChatMessageV3». В таблице 1 представлены интересующие специалистов кибербезопасности поля данного объекта с пояснением, какие данные о сообщении они хранят.

Номер поля в объекте может меняться в зависимости от версии приложения Snapchat в связи с добавлением дополнительных полей в более поздних версиях приложения. При этом сохраняется порядок полей относительно друг друга. Для определения позиции поля, хранящего данные в списках объектов или объектах некоторых классов, можно применять метод аналогичный способу поиска массива сообщений в объекте диалога. К сожалению, не все поля, хранящие интересующую специали-

```
1) информация о приложенном стикере
2) информация о приложенных медиафайлах
3) местоположение
4) имя отправителя
5) время отправки, клиентское
6) статус сохранения сообщения у клиентов
7) время отправки, серверное
8) текст сообщения
```

Рис. 3: Относительный порядок расположения полей в объекте, хранящем информацию о сообщении

стов кибербезопасности информацию, имеют объектный тип, например, поле, хранящее текст сообщения, является строкой, а поле, содержащее время отправки — числом с плавающей запятой двойной точности или указателем на ранее упоминаемое в файле число с таким же типом данных. В связи с этим процесс извлечения данных о сообщении состоит из двух этапов:

1. Определить позиции полей, уникально идентифицируемых по типу (списки объектов или объекты, реализующие некоторый класс, которые встречаются среди списка всех полей сообщения в единственном экземпляре)

2. Определить позиции оставшихся полей, хранящих интересующую специалистов кибербезопасности информацию, путем поиска их по типу значений в промежутке позиций, ограниченном полями с уже определенным положением.

Для реализации такого алгоритма извлечения данных был составлен относительный порядок расположения полей в объекте, хранящем информацию о сообщении, представленный на рис. 3.

После определения позиции полей в объекте необходимо извлечь хранящуюся в них информацию. Для полей, имеющих примитивный тип данных, данный процесс является тривиальным, однако поля, хранящие информацию о медиафайлах, стикерах и геопозиции, требуют дополнительного анализа.

4.3. Извлечение медиафайлов

Информация о медиафайлах может храниться как в виде объекта (если медиафайл в сообщении один), так и в виде массива объектов, реализующих класс «SCChatMediaContent».

При анализе данного объекта было выявлено, что он хранит строку — ключ из базы данных sqlite, хранящейся в файле «SnapchatFolder/Documents/user_scoped/contentManagerV3/contentManagerDb.db», в одном из своих полей (см. рис 4).

После извлечения ключа из объекта, хранящего информацию о медиафайле, необходимо получить имя данного медиафайла из базы данных по извлеченному ключу.

В ходе исследования было выявлено, что медиафайлы из диалогов приложение Snapchat сохраняет в папке «SnapchatFolder/Documents/com.snap.file_manager_3_SCContent_<id пользователя> », где <id пользователя> — уникальный идентификатор пользователя, который хранится в файле «chatConversationStore.plist». При этом файлы хранятся без расширения, поэтому для их просмотра стандартными средствами необходимо определить конкретный тип файла, используя заголовки известных форматов хранения мультимедиа в файлах.

4.4. Извлечение стикеров

Информация о стикере, прикрепленном к сообщению, сохраняется в объекте, реализующем класс «SCChatSticker». В данном объекте (см. рис. 5) особый интерес представляют три поля, имеющие строковый тип — определяющие вид стикера (стикер, эмоджи или битмоджи), стикерпак, из которого был взят данный стикер, и имя стикера (unicode код для эмоджи).

Рис. 4: Объект, хранящий информацию о приложенном медиафайле. Красным выделено поле, хранящее ключ из базы данных медиафайлов.

4.5. Извлечение геопозиции

Информация о геопозиции хранится в объекте, реализующем класс «SCChatMessageParcel» в поле, хранящем данные в необработанном виде. Данные о геолокации представлены в формате json, показанном на рис. 6. В полях «sender_lat» и «sender_lng» хранятся координаты отправителя в момент последнего обновления диалога на устройстве, из памяти которого был извлечен исследумый файл.

Рис. 5: Объект, хранящий информацию о приложенном стикере. Красным выделены поля, хранящие имя стикерпака, название стикера и его вид сверху вниз соответсвенно.


```
{
  "recipient_user_id": "da31b228-7c62-4108-96bc-bc52c0e9e8c5",
  "sender_lat": 59.88172948831681,
  "sender_user_id": "cd3417a8-c57d-42f7-90ed-58053732a427",
  "sender_lng": 30.2748246398105,
  "user_response": 0
}
```


Рис. 6: Местоположение пользователя, сохраненное в формате JSON внутри одного из полей объекта, хранящего информацию о геопозиции.

5. Реализация

Для извлечения данных приложения Snapchat из файла формата TSAF был реализован прототип на языке C#, выбранном для упрощения будущей интеграции разработанного решения в основной продукт компании Belkasoft. Данный прототип позволяет прочитать и распарсить файл формата TSAF, а затем извлечь из него информацию об имени пользователя и его диалогах.

5.1. Архитектура

Реализованный прототип представляет собой конвейер (см. рис. 7), в котором на вход подается файл «chatConversationStore.plist», имеющий формат TSAF. В качестве первого этапа данный файл обрабатывается парсером и строится его представление в памяти компьютера, затем данное представление передается анализатору, который извлекает из него все необходимые специалистам кибербезопасности данные. В заверешении извлеченные данные передаются в модуль, отвечающий за вывод информации, который упаковывает полученную информацию в zip архив.

Рис. 7: Архитектура реализованного прототипа

5.2. Парсер

Для разбора файла формата TSAF был реализован парсер, который побайтово считывает переданный ему файл и строит представление хранимых в нем данных в виде дерева. Для этого используется иерархия классов, представляющих тот или иной тип данных в файле формата TSAF. Данная иерархия представлена на рис. 8.

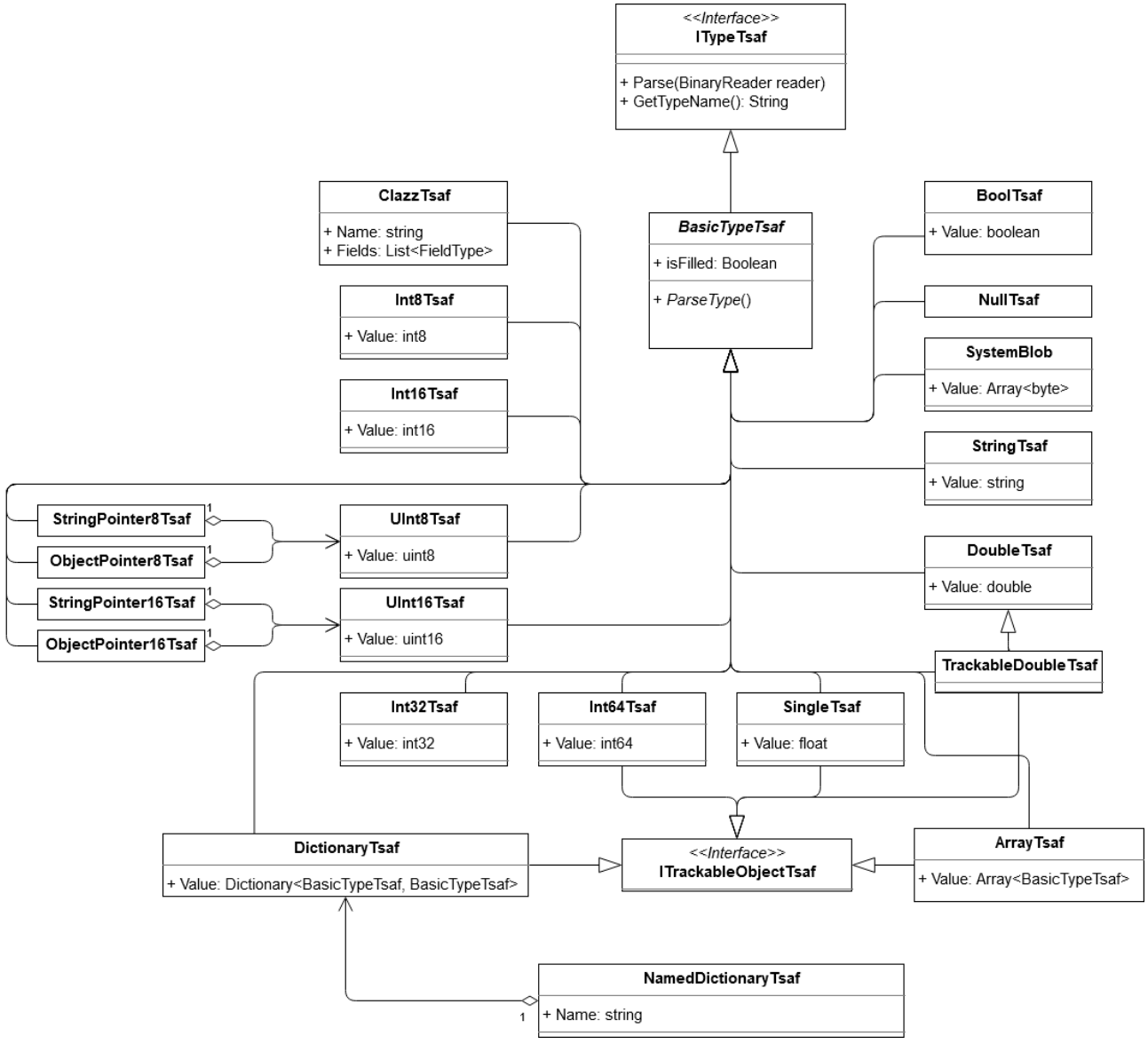

Рис. 8: Иерархия классов для парсинга файла формата TSAF

Для определения структуры и типа данных каждого поля используются приведенные выше результаты исследования данного формата. Так же парсер отвечает за трансляцию указателей в значения, на ко-

торые они указывают. Для этого все данные, на которые могут указывать указатели, помещаются в специальные списки, позицию в которых и хранит в себе указатель в формате TSAF. Для пометки типов данных, которые необходимо заносить в данные списки, используется интерфейс «ITrackableObject».

5.3. Анализ и вывод извлеченных данных

Построенное парсером представление данных, содержащихся в файле формата TSAF, передается специальному модулю для анализа, чтобы получить информацию о диалогах пользователя приложения Snapchat. Данный модуль извлекает имя пользователя и массив сообщений, а затем для каждого сообщения собирает всю доступную в файле «chatConversationStore.plist» информацию, используя алгоритм, описанный выше. Для представления извлеченной информации используется набор вложенных классов, представленный на рис 9.

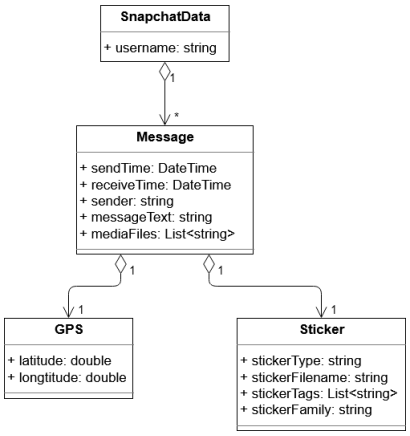

Рис. 9: Классы, использующиеся для представления извлеченной анализатором из файла «chatConversationStore.plist» информации

Следующим шагом работы прототипа является упаковка извлеченной информации в zip архив, структура которого показана на рис. 10.

Для этого объект класса «SnapchatData», полученный от анализатора, сохраняется в формате JSON в файл «result.json». Дополнительно в архив из папки приложения Snapchat копируются медиафайлы и стикеры для возможности их просмотра при изучении данных, сохраненных в файл «result.json».

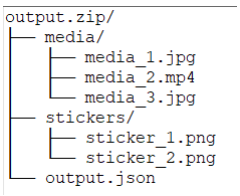

Рис. 10: Структура архива, который является результатом работы прототипа

5.4. Апробация

Для проверки корректности работы прототипа была проведена апробация с использованием тестового смартфона iPhone 5S. В ходе апробации из памяти данного смартфона была извлечена папка приложения Snapchat, путь к которой был передан прототипу для получения информации о диалогах пользователя. Для результатов работы прототипа была проведена верификация, подтвердившая полное совпадение извлеченной информации и инфомации, доступной на тестовом устройстве в приложении Snapchat.

Заключение

В ходе данной работы было выявлено отсутствие инструментов и подробных описаний для формата хранения данных TSAF. Для изучения внутреннего устройства файлов формата TSAF на тестовом смартфоне были поставлены эксперименты с использованием приложения Snapchat, которые позволили определить схему хранения данных приложения Snapchat в файлах формата TSAF. Полученная в ходе исследования информация позволила описать внутреннее устройство файлов данного формата и реализовать прототип парсера для файлов формата TSAF.

Для извлечения информации об имени пользователя и его диалогах в приложении Snapchat был реализован прототип, способный читать файлы формата TSAF, в которых хранится информация о сообщениях в данном приложении, и извлекать из них интересующую специалистов кибербезопасности информацию. Прототип был апробирован на тестовом смартфоне.

Таким образом, в ходе данной работы были выполнены следующие задачи:

1. исследован новый формат данных TSAF и сделан его обзор;
2. реализован прототип парсера файла TSAF;
3. разработан прототип для извлечения сообщений приложения Snapchat из снимка памяти iOS;
4. апробирован реализованный прототип на тестовом смартфоне iPhone 5S.

Список литературы

[1] Apple Inc. About Property Lists // официальный сайт. — Access mode: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/PropertyLists/AboutPropertyLists/AboutPropertyLists.html#//apple_ref/doc/uid/10000048i-CH3-SW2 (online; accessed: 12.12.2019).

[2] Apple Inc. Binary Plist // официальный сайт. — Access mode: https://opensource.apple.com/source/CF/CF-550/CFBinaryPList.c (online; accessed: 12.12.2019).

[3] Apple Inc. Understanding XML Property Lists // официальный сайт. — Access mode: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/PropertyLists/UnderstandXMLPlist/UnderstandXMLPlist.html#//apple_ref/doc/uid/10000048i-CH6-SW1 (online; accessed: 12.12.2019).

[4] Belkasoft // официальный сайт. — Access mode: https://belkasoft.com (online; accessed: 27.10.2020).

[5] Belkasoft. Belkasoft Evidence Center // официальный сайт. — Access mode: https://belkasoft.com/ec (online; accessed: 27.10.2020).

[6] DoubleBlak Digital Forensics. SnapChat PList // официальный сайт. — Access mode: https://www.doubleblak.com/blogPosts.php?id=5 (online; accessed: 18.12.2019).

[7] DoubleBlak Digital Forensics. SnapChat PList // официальный сайт. — Access mode: http://doubleblak.com/blogPosts.php?id=5 (online; accessed: 27.10.2020).

[8] DoubleBlak Digital Forensics. Spoopy // официальный сайт. — Access mode: https://doubleblak.com/software.php (online; accessed: 12.12.2019).

[9] SQLite // официальный сайт. — Access mode: https://www.sqlite.org/index.html (online; accessed: 27.10.2020).

[10] Snap Inc. Snapchat // официальный сайт. — Access mode: https://snapchat.com (online; accessed: 12.12.2019).

[11] Synchronization Cellebrite Mobile. Cellebrite // официальный сайт. — Access mode: https://cf-media.cellebrite.com/wp-content/uploads/2019/01/ReleaseNotes_UFED_UFEDPA_7.13.pdf (online; accessed: 12.12.2019).

[12] Whiffin Ian. About DoubleBlak Digital Forensics // официальный сайт. — Access mode: https://www.doubleblak.com/index.php (online; accessed: 18.12.2019).

[13] Виноградов Михаил. Исследование уязвимости iOS tfp0 для применения в криминалистическом анализе // дипломная работа. — online; accessed: http://se.math.spbu.ru/SE/diploma/2019/bmo/441-Vinogradov-report.pdf (online; accessed: 18.12.2019).