Санкт-Петербургский государственный университет
Математическое обеспечение и администрирование информационных систем
Информационные системы и базы данных

Виноградов Михаил Викторович

Исследование уязвимости iOS tfp0 для применения в криминалистическом анализе

Выпускная квалификационная работа

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

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

Рецензент:
специалист лаборатории компьютерной криминалистики и анализа вредоносного кода
ООО "Группа АйБи" Михайлов И. Ю

Санкт-Петербург
2019

SAINT-PETERSBURG STATE UNIVERSITY

Information Systems Administration and Mathematical Support Software Engineering

Vinogradov Mikhail

Application of iOS tfp0 vulnerability for IT forensic analysis

Graduation Project

Scientific supervisor:

Candidate of Engineering Sciences Yuri Litvinov

Consultant:

Head of software development Belkasoft Nikita Timofeev

Reviewer:

Digital forensics analyst Group-IB Igor Mikhaylov

Saint-Petersburg
2019

Оглавление

Введение

В современном мире невозможно представить себе жизнь без компьютеров и мобильных устройств. Поэтому данные, хранящиеся на них, важны правоохранительным органам для раскрытия или предотвращения различных преступлений. В исследовании этих устройств помогает компьютерная криминалистика.

Компьютерная криминалистика или форензика – прикладная наука о раскрытии преступлений, связанных с цифровой информацией, об исследовании цифровых доказательств, методах поиска, получения и закрепления таких доказательств [1]. Одним из важных составляющих данной науки является анализ мобильных телефонов и планшетов, поскольку именно они являются основными и наиболее используемыми устройствами для большинства людей в современном мире.

Особый интерес у криминалистов вызывает анализ мобильных устройств компании Apple – iPhone и iPad. Данный интерес связан прежде всего с закрытостью установленной на них операционной системы iOS, которая заключается в отсутствии прямого доступа к файловой системе. Данная проблема существенно затрудняет анализ информации на них или делает его вовсе невозможным. Из файловой системы устройства может быть извлечено огромное количество информации, недоступной обычному пользователю. Примером такой информации могут служить хранящиеся геолокационные данные вышеуказанных устройств для составления значимых мест в приложениях «Карты», «Календарь», «Фото» и других программах [2]. Геоданные за последние несколько дней с некоторой периодичностью по времени сохраняются в специальную базу, которую невозможно прочитать обычному пользователю. Из этих данных можно узнать, посещал ли подозреваемый место преступления и т. д.

В целях анализа данных, позволяющих определить местонахождение пользователя устройства, то есть проведения цифровой криминалистической экспертизы, разработано программное обеспечение Belkasoft Evidence Center.

Оно позволяет анализировать данные на мобильных устройствах, жестких дисках, облачных сервисах и т. д. Вместе с тем, для анализа всех данных приложений, установленных на устройствах под управлением операционной системы iOS, при помощи данного инструмента требуется снятие полного образа файловой системы [3]. Однако данная функциональность отсутствует в Belkasoft Evidence Center, что, в свою очередь, усложняет проведение анализа мобильного устройства.

Постановка задачи

Целью данной работы является разработка программы для создания образа файловой системы устройств на операционной системе iOS и внедрение ее в коммерческий продукт Belkasoft Evidence Center. Для достижения цели были поставлены следующие задачи:

- исследовать способы получения данных из мобильного устройства на операционной системе iOS;
- исследовать суть уязвимости «tfp0» и возможности применения данной уязвимости для получения доступа к данным на мобильном устройстве;
- реализовать приложение для взаимодействия между устройством и персональным компьютером;
- реализовать приложение-прототип для снятия образа и внедрить его в Belkasoft Evidence Center;
- провести апробацию полученного результата.

1. Обзор

1.1. iOS

iOS – операционная система, созданная компанией Apple в 2007 г., которая устанавливается исключительно на серии смартфонов Apple iPhone, музыкальных плееров iPod Touch и планшетов Apple iPad. В ее основе лежит операционная система OS X. Данная ОС является закрытой, что проявляется в отсутствии у пользователя прямого доступа к файловой системе и возможности установки сторонних приложений не из официального магазина App Store. Однако запуск и установка сторонних приложений возможны после проведения процедуры Jailbreak. По состоянию на 01.04.2019 последняя версия операционной системы – iOS 12.2, выпущенная 25 марта 2018 года [4].

1.2. Jailbreak

Jailbreak – это процедура по получению полных прав на модификацию файлов в системных папках iOS, аналогичная получению прав пользователя root в UNIX-образных системах [5]. Данная процедура позволяет пользователю получить возможность доступа к файловой системе и установке неподписанных приложений[1]. Основным источником таких программ является приложение Cydia, которое представляет собой магазин приложений для устройств на базе iOS, прошедших процедуру Jailbreak.

Существует три вида Jailbreak:

- привязанный – полные права теряются при перезагрузке устройства. Однако устройство становится неработоспособным, пока не будет проведена повторная процедура Jailbreak. Для последних версий iOS данного вида Jailbreak не существует;
- непривязанный – все права сохраняются, даже при перезагрузке;

1 Неподписанные приложения – это приложения, которые не подписаны сертификатом безопасности Apple

- полунепривязанный – полные права теряются при перезагрузке устройства, однако в отличие от привязанного Jailbreak устройство не теряет работоспособность при перезагрузке. Для повторного получения полных прав необходимо повторить процедуру Jaibreak. Именно данный тип используется для взлома современных версий iOS.

Мобильные устройства с проведенной процедурой Jailbreak лишаются гарантийных обязательств и прав на техническую поддержку. Стоит отметить, что при установке операционной системы заново, через обновление или восстановление, все следы от процедуры Jailbreak удаляются, что, в свою очередь, возвращает гарантию. В США действия по получению полных прав не являются противозаконными [6].

1.2.1 Примеры программ для проведения процедуры Jailbreak

1. Electra – для устройств на базе iOS 11.0 – 12.1.2.
2. RootlessJB – для iOS 12 – 12.1.2.
3. Unc0ver – для устройств на iOS 11.0 – 11.4.
4. Yalu102 – iOS 10.0 – 10.3 [7].

1.3. Основные виды извлечения данных с устройств на базе iOS

Для извлечения данных с устройств компании Apple на практике существует два способа.

- Первый способ – снятие логического образа устройства, при извлечении которого используются стандартные службы и сервисы телефона, вызывающиеся стандартными программами iTunes[2] или Xcode[3]. Примером извлечения логического образа

[2] iTunes — медиаплеер для организации и воспроизведения музыки и фильмов, разработанный компанией Apple и бесплатно распространяемый для платформ macOS и Windows

[3] Xcode — интегрированная среда разработки (IDE) программного обеспечения для платформ macOS, iOS, watchOS и tvOS, разработанная корпорацией Apple.

служит создание резервной копии через программу iTunes, для чего не нужно устанавливать на устройство никаких дополнительных программ. В этом случае при анализе резервной копии можно извлечь некоторую полезную информацию о пользователе: список контактов, историю сообщений в мессенджере WhatsApp, SMS-сообщения, фото, видео и другие типы данных.

- Второй способ – снятие физического образа. Данный способ позволяет получить полный шифрованный побитовый образ файловой системы. Ключевой особенностью для физического извлечения данных является наличие полных прав у пользователя (root пользователь). Из данных, полученных этим путем, извлекается гораздо больше информации о владельце устройства: дополнительно к данным, получаемым при использовании первого способа, скрытые геоданные пользователя, удаленные сообщения, пароли и т. д.

1.4. Полный логический образ файловой системы

При работе с физическими образами возникает существенная проблема – это их зашифрованность. К настоящему времени не найдено способа расшифрования образа, поэтому он является бесполезным для криминалиста. Но, вместе с тем, существует способ снятия данных, при котором извлекается значительно больше информации, чем при обычном логическом извлечении данных. Данный способ называется «снятие полного логического образа файловой системы». Он возможен только при наличии проведенной процедуры Jailbreak. При извлечении образа, помимо стандартных служб программы iTunes, также используются возможности дополнительных пакетов из магазина Cydia. Важными отличиями полного логического образа от простого является то, что, во-первых, файлы извлекаются в явном виде, то есть, к примеру, как при простой передаче файлов между usb накопителем и персональным компьютером, в то время, как при простом – извлекаемые

данные хранятся в специальном формате с головной базой Manifest.db, во-вторых, объем извлекаемых данных при полном образе значительно больше, так как извлекается огромное количество системных баз данных, локальных баз данных приложений, их файлы журналов и т. д.

1.5. Протокол SSH

Для того чтобы проводить удаленное управление операционной системой и туннелирование TCP-соединений, используемое, к примеру, для передачи файлов, применяется протокол SSH [8]. Он обеспечивает безопасный вход в удаленную систему и создание других безопасных служб в сетях, не обеспечивающих безопасность. На рисунке 1 представлен упрощенный процесс настройки безопасного соединения между клиентом и сервером:

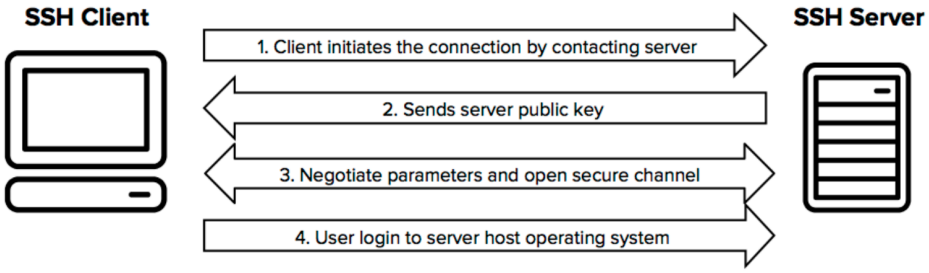

Рис 1. Процесс настройки безопасного подключения[4].

Протокол SSH состоит из трех основных компонент:

- протокол транспортного уровня (Transport Layer Protocol), который обеспечивает аутентификацию, конфиденциальность и целостность серверов;
- протокол аутентификации пользователей (User Authentication Protocol), используемый на серверах для проверки полномочий клиентов;

[4] SSH Protocol. – URL: https://www.ssh.com/ssh/protocol/

- протокол соединений (Connection Protocol), обеспечивающий мультиплексирование шифрованного туннеля в несколько логических каналов [9].

Последний протокол особенно полезен для взаимодействия между устройством на операционной системе iOS и компьютером, поскольку в данном случае поддерживается перенаправление протоколов типа SMTP, POP3 и HTTP. Таким образом, посредством протокола SSH создается туннель, в котором данные на одном конце соединения шифруются и расшифровываются на другом.

После проведения процедуры Jailbreak для установки SSH соединения с устройством появляется возможность установки специального пакета OpenSSH, который является открытой версией протокола SSH от независимого разработчика Сэма Бингера

1.6. Существующие решения

1.6.1 Elcomsoft iOS Forensic Toolkit

Elcomsoft iOS Forensic Toolkit – платный продукт для криминалистического исследования устройств на основе Apple iOS. Данный инструмент позволяет проводить правоохранительным органам судебные и компьютерно-технические экспертизы устройств iPhone, iPad и iPod производства компании Apple, работающих под управлением iOS версий с 7 по 12. Данная утилита предоставляет возможность создать полный физический образ файловой системы, логический образ, расшифровать пароли, коды и другую защищенную информацию. Основным требованием для большей части функциональности является наличие Jailbreak [10].

1.6.2 Oxygen Forensic Extractor

Oxygen Forensic Extractor – программное обеспечение, созданное компанией Oxygen Forensic. Данная утилита предлагает как логические, так и физические методы получения данных с устройства через обычный USB-кабель. Программа поддерживает устройства под управлением iOS, Android,

Windows Phone, Windows Mobile, Blackberry, Bada, Symbian OS или вообще не имеющих ОС (функциональные телефоны). Также доступна поддержка китайских чипсетов MTK и Spreadtrum [11].

2. Уязвимость tfp0

Процедура Jailbreak представляет собой использование набора уязвимостей, чтобы преодолеть защитные механизмы Apple. Рассмотрим далее в обобщённом виде её основные этапы.

Основополагающим этапом является получение доступа на чтение и запись к сегменту виртуальной памяти ядра[5]. Для этого необходимо получить ссылку на этот сегмент, которая называется порт задачи ядра. В операционных системах семейства OS X task_for_pid – это функция, которая позволяет привилегированному процессу получать порт задачи другого процесса на том же хосте, но при этом на операционной системе iOS получение порта для задачи ядра блокируется из-за ограничений доступа. Для обхода ограничения используются уже известные уязвимости или находятся новые, позволяющие выполнять произвольный код в привилегированном контексте. На их основе создается специальный патч, разрешающий любой исполняемой программе работать от имени суперпользователя для вызова task_for_pid для задачи ядра, которая имеет идентификатор процесса[6] 0. Из этого всего и происходит название tfp0, то есть task_for_pid 0. Все эти действия дают доступ для вызова функций чтения и записи в виртуальную память, которые называются vm_read и vm_write соответственно, и их дальнейшего использования для модификации области виртуальной памяти ядра. Примером может служить патч, используемый в последних Jailbreak, на основе ошибки CVE-2019-6225 [12], которая заключается в уязвимости подсчета ссылок в функции task_swap_mach_voucher. Он создает поддельный порт задачи ядра, что позволяет записывать и читать произвольную память ядра.

Следующим важным этапом является получение root доступа. Самый распространенный способ получить root в современных jailbreak – это

[5] Ядро – центральная часть операционной системы (ОС), обеспечивающая приложениям координированный доступ к ресурсам компьютера, таким как процессорное время, память, внешнее аппаратное обеспечение, внешнее устройство ввода и вывода информации.
[6] Индентификатор процесса – уникальный номер (идентификатор) процесса в многозадачной операционной системе.

соединить учетные данные процесса приложения Jailbreak с ядром. Он дает возможность работать приложению как приложение с наивысшими правами. В iOS 11 и 12 данный метод получения прав не вызывает паники ядра, однако в других версиях она возможна. Паника ядра вызывается специальным механизмом защиты Kernel Patch Protection при проверке целостности ядра через случайные промежутки времени, если есть подозрения, что ядро подверглось изменениям.

Наличие root прав недостаточно для получения доступа к файловой системе. Причиной этому служит мера безопасности – sandbox или песочница. У приложения есть доступ только к собственным данным. Для выхода за пределы контейнера в большинстве современных версий iOS необходимо передавать полномочия ядра через копирование в процесс, которому требуется выход за пределы песочницы. Другой способ выхода за пределы – это аннулирование в полномочиях данного пользователя специального указателя Mandatory Access Control (обязательный контроль доступа). В последних версиях Jailbreak используется последний способ.

Следующая стадия Jailbreak заключается в том, чтобы позволить телефону запускать свои двоичные приложения. Этому препятствует специальное расширение ядра AppleMobileFileIntegrity (AMFI), которое обеспечивает целостность кода, работающего в операционной системе, и является основой для проверки подписи кода. Данное расширение необходимо подвергнуть изменениям, которые заключаются в запуске «поддельных» двоичных файлов. В этих целях следует изменить существующие у AMFI проверки, для чего создается динамическая библиотека, заменяющая функцию проверки пользовательского кода и вставляющая ее в AMFI. Для ускорения Apple кэширует хэши сигнатур каждого двоичного файла Apple iOS в так называемые «trustcache». Если ядро видит, что хэш находится в кэше доверия, оно не выполняет дальнейшую проверку, а просто предполагает, что двоичный файл является «доверенным» и разрешает выполнение кода. Таким образом

находится порт задачи AMFI для дальнейшего исправления. Затем созданная библиотека добавляется в специальный trustcache.

На последнем этапе следует установить бинарные файлы, такие как tar, ssh, dpkg и т. д. Для этого необходимо перемонтировать на чтение и запись корневой раздел «/», который по умолчанию монтируется только для чтения. Обычно это делается с помощью комбинации исправлений ядра и стандартных системных функций. Однако в последней, двенадцатой версии iOS существует способ установки бинарных файлов без перемонтирования файловой системы, который реализуется путем достижения приложением определенных прав, подписания бинарных файлов дополнительным параметром «com.apple.private.security.container-required» со значением false и установкой их по пути «/var/containers/Bundle/».

В рамках данной работы были исследованы и использованы существующие реализации перечисленных выше шагов атаки на iOS.

3. Архитектура прототипа

Реализуемая система работает в несколько этапов. Сначала происходит туннелирование TCP-соединения с локального порта на компьютере на 22 порт (SSH порт) [13] на устройстве. Затем создается безопасное SSH-соединение для беспарольного доступа к устройству. Далее происходит отключение автоблокировки устройства. И последний этап – снятие самого полного логического образа файловой системы. Опишем все этапы более подробно.

3.1 Туннелирование TCP-соединения по USB

Для подключения к устройству используется OpenSSH. Возможны два способа взаимодействия. Первый – подключение через Wi-Fi соединение, однако такое соединение является недостаточно стабильным. Второй способ взаимодействия iOS устройства с компьютером, которое обеспечивает стабильное соединение – через USB. Оно возможно только при установленной программе iTunes, которая использует специальный драйвер для взаимодействия. Следовательно, все общение с устройством реализуется посредством сервисов программы iTunes.

Для начала работы с сервисами необходимо установить сокет-соединение[7] с протоколом типа TCP на адрес 127.0.0.1, что соответствует localhost, и портом 27015. Порт 27015 на компьютере на базе Windows использует сервис AppleMobileDeviceService, который отвечает за распознавание устройств iPhone, iPad и iPod touch в программе iTunes [14]. Таким образом, ошибка при подключении к данному порту свидетельствует об отсутствии программы iTunes на компьютере. Для дальнейшего подключения необходимо получить список устройств. Для этого отправляется запрос через открытый ранее сокет. Данные для запроса формируются в формат «plist» – это специальный формат на основе языка разметки «XML», где список параметров представлен в виде пар «Ключ – Значение» [15]. В нем

[7] Сокет – название программного интерфейса для обеспечения обмена данными между процессами.

прописывается идентификатор приложения (Bundle ID), дается его описание, указывается тип сообщения (в данном случае указывается «ListDevices», что соответствует списку устройств) и имя программы. Пример данных для запроса можно увидеть на рисунке 2.

В ответ на запрос от устройства приходит сообщение в формате «plist» (представлено на рисунке 3). В нем содержится ключ «DeviceID», который обозначает идентификатор устройства на данном компьютере, и список параметров устройства, среди которых есть его серийный номер.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>BundleID</key>
        <string>org.myapp.usbmuxd</string>
        <key>ClientVersionString</key>
        <string>built for freedom</string>
        <key>MessageType</key>
        <string>ListDevices</string>
        <key>ProgName</key>
        <string>myusbapp</string>
        <key>kLibUSBMuxVersion</key>
        <integer>3</integer>
</dict>
</plist>
```

Рис 2. Plist для получения списка устройств.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>DeviceList</key>
        <array>
                <dict>
                        <key>DeviceID</key>
                        <integer>46</integer>
                        <key>MessageType</key>
                        <string>Attached</string>
                        <key>Properties</key>
                        <dict>
                                <key>ConnectionType</key>
                                <string>USB</string>
                                <key>DeviceID</key>
                                <integer>46</integer>
                                <key>LocationID</key>
                                <integer>0</integer>
                                <key>ProductID</key>
                                <integer>4776</integer>
                                <key>SerialNumber</key>
                                <string>c9aab3959af3ea32e85888d2885fd463e5bc9ca0</string>
                        </dict>
                </dict>
        </array>
</dict>
</plist>
```

Рис 3. Plist со списком устройств.

На следующем этапе организуется непосредственное подключение для передачи данных между iOS устройством и компьютером. Для этого необходимо снова отправить сообщение в формате «plist» на устройство. В нем снова указывается имя приложения, тип сообщения, а также указывается тип «Connect» для подключения устройства, «DeviceID», полученный ранее, и «PortNumber», в котором указывается порт на устройстве, к которому необходимо подключение. Конечный вид запроса представлен на рисунке 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>BundleID</key>
        <string>org.myapp.usbmuxd</string>
        <key>ClientVersionString</key>
        <string>built for freedom</string>
        <key>MessageType</key>
        <string>Connect</string>
        <key>ProgName</key>
        <string>myusbapp</string>
        <key>kLibUSBMuxVersion</key>
        <integer>3</integer>
        <key>DeviceID</key>
        <integer>46</integer>
        <key>PortNumber</key>
        <integer>22</integer>
</dict>
</plist>
```

Рис 4. Plist для создания подключения.

В ответ на запрос от устройства приходит сообщение, представленное на рисунке 5. Чтобы понять, успешно ли прошло соединение, необходимо посмотреть на значение ключа «Number»: если подключение разрешено, то значение должно быть рано 0.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>MessageType</key>
        <string>Result</string>
        <key>Number</key>
        <integer>0</integer>
</dict>
</plist>
```

Рис 5. Plist со статусом подключения.

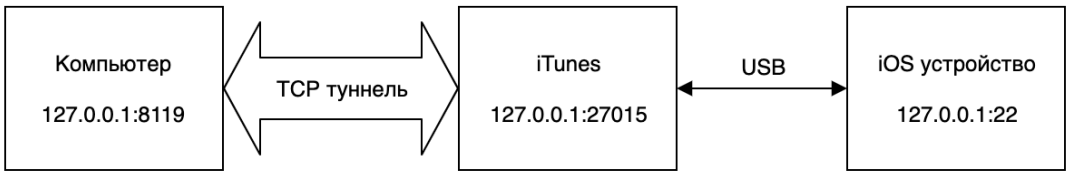

Рис 6. TCP-туннель между компьютером и устройством.

Заключительным этапом подключения является создание TCP-туннеля, схема которого представлена на рисунке 6. В качестве хоста выступает localhost и его свободный порт на локальном компьютере. Сервером выступает созданный ранее сокет, подключенный к порту 27015. Между хостом и сервером идет непрерывная передача данных через буфер. Таким образом, данные, отправленные с хоста, будут направлены напрямую на 22 порт на iOS устройстве и в обратном порядке.

3.2 Схема создания беспарольного взаимодействия компьютера с устройством по OpenSSH

Ранее в работе отмечалось, что после проведения процедуры Jailbreak появляется возможность установки пакета OpenSSH.

Для удобства использования OpenSSH в Belkasoft Evidence Center необходимо создать беспарольное соединение, для чего необходимо выполнить следующие стандартные шаги, которые представлены на рисунке 7.

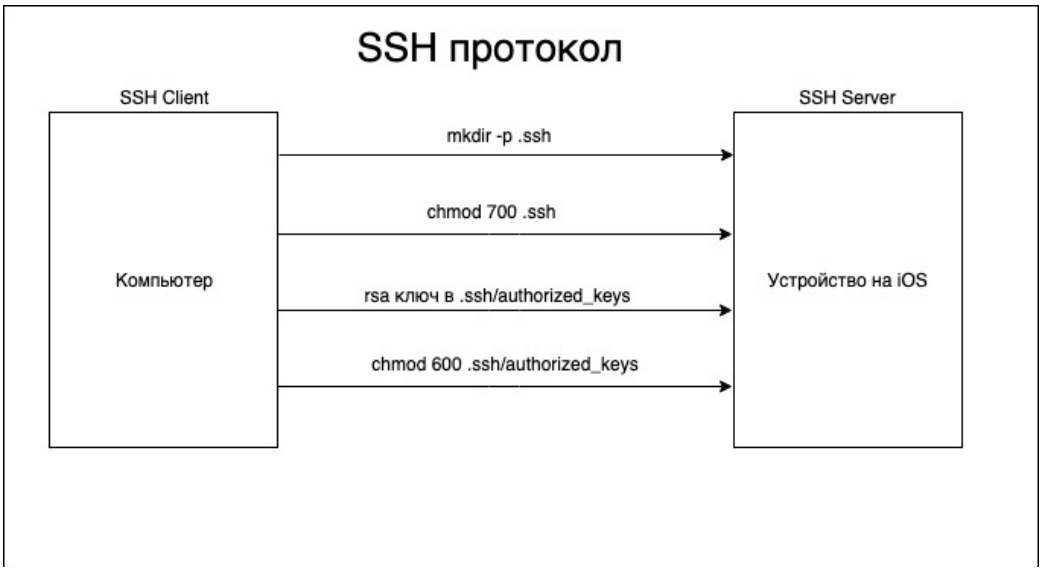

Рис 7. Схема создания беспарольного соединения.

В первую очередь, на локальном компьютере с помощью утилиты ssh-keygen создается rsa-ключ размером 2048 бит. Генерируется две версии ключа: для локального компьютера и для ssh-сервера с расширением «pub». Далее открывается TCP-тоннель, описанный в предыдущем пункте. Затем с помощью специального пакета Renci.SSH для использования SSH при создании программ на языке C# осуществляется подключение к устройству с использованием стандартной связки логина «root» и пароля «alpine», где создается папка «.ssh». Данной папке выставляется полный доступ только для root-пользователя. С помощью SCP[8] ключ загружается на устройство по пути .ssh/authorized_keys. Затем файлу выставляются доступ на чтение и запись только для root-пользователя.

3.3 Принцип создания полного логического образа

Далее рассмотрим заключительный этап – снятие полного логического образа файловой системы. Так как было создано подключение, основанное на rsa-ключе, то при использовании OpenSSH пароль больше не требуется. Для создания образа, в первую очередь, будет использоваться утилита ssh.exe пакета OpenSSH для Windows.

[8] SCP (Secure CoPy) — программа для удаленного копирования фалов по сети между хостами.

Операционная система iOS является UNIX-образной системой, поэтому для нее применимы стандартные утилиты командной строки. Одной из них является утилита «tar», которая представляет собой стандартный архиватор в UNIX-системах. Утилита автоматически устанавливается в процессе проведения процедуры Jailbreak. В создании образа помогает «Pipe» или «именованный канал» – один из методов межпроцессного взаимодействия, расширение понятия конвейера[9] в Unix и подобных ОС. Именованный канал позволяет различным процессам обмениваться данными, даже если программы, выполняющиеся в этих процессах, изначально не были спроектированы для взаимодействия с другими программами. Сначала происходит вызов «ssh.exe -p 8119 -i {путь до rsa-ключа} root@127.0.0.1 tar -cf - /», где:

- «-p» соответствует порту хоста в созданном до устройства TCP-туннеле;
- «root» – имя пользователя;
- «-cf» означает создать архив из указанного места без сжатия, «/» папка, на основе которой будет создан архив, в данном случае это главная директория;
- «-», который означает, что архив не сохраняется ни в какой файл, а все данные напрямую будут выведены.

Затем открывается именованный канал, который все данные, полученные напрямую, будет перенаправлять в новый файл на локальном компьютере.

3.3.1 Анализ целесообразности сжатия образа

В большинстве случаев основополагающим фактором для проведения криминалистической экспертизы является время снятия образа, поэтому встал вопрос о том, возможно ли использовать сжатие архива для уменьшения времени создания образа. После проведения процедуры Jailbreak tar-образ на

[9] Конвейер (UNIX) – перенаправление вывода одного программного процесса на ввод другого процесса.

устройстве возможно сжимать с помощью следующих утилит, работающих совместно с утилитой «tar»:

- gzip – утилита сжатия и восстановления (декомпрессии) файлов, использующая алгоритм Deflate[10]. С «tar» она взаимодействует через специальный флаг «-z»;
- lz4 – утилита, использующая чрезвычайно быстрый алгоритм сжатия lz4[11]. В отличие от gzip, она используется через «pipe» вызова «tar».

Для того чтобы понять целесообразность применения представленных архиваторов, был проведен сравнительный анализ по времени получения, скорости загрузки и итоговому размеру образа с iPhone 6S, подключенному к компьютеру через USB 3.0, со сжатием и без сжатия. Было создано пять образов для каждого варианта сжатия, для которых время загрузки и конечный размер совпадали с точностью до единиц секунд и до мегабайт соответственно.

	lz4	**gzip**	**Без сжатия**
Время (мин)	14	27	6,5
Скорость (Мб/с)	13	3	27,6
Размер (Мб)	6270	5007	10800

Таблица 1 Сравнительный анализ создания образа с архивацией и без

Как видно из анализа, представленного в таблице 1, различные алгоритмы сжатия проигрывают по времени и скорости загрузки созданию образа без сжатия. Например, разница по времени между lz4 и без сжатия составила более чем в 2 раза.

В результате сравнительного анализа на реальных устройствах было установлено, что для того чтобы достичь минимального времени снятия

[10] Deflate — это алгоритм сжатия без потерь, использующий комбинацию алгоритмов LZ77 и Хаффмана.

[11] LZ4 — алгоритм сжатия данных без потерь, относящийся к семейству методов сжатия LZ77, работающих с байтовыми потоками.

образа с iOS устройства, необходимо использовать утилиту «tar» без дополнительных утилит сжатия. Данный результат обусловлен тем, что большую часть памяти устройства, как правило, занимают фотографии, видео и прочий контент, который уже хранится в сжатом формате. Из этого следует, что процессор дополнительно нагружается на сжатие разных системных файлов, которые не несут в себе ценной информации для криминалиста. Именно поэтому происходит увеличение времени создания образа, а не уменьшение.

3.4 Автоматическое снятие автоблокировки экрана

3.4.1 Общее описание подхода

Существует одна особенность, которая мешает при создании полного логического образа, способом, представленным выше. На реальных устройствах у всех пользователей стоит автоблокировка экрана. Это приводит к тому, что во время создания образа устройство будет автоматически блокироваться. В этом случае не все данные будут загружены с устройства. Это связано с тем, что приложения могут сохранять свои файлы с использованием специального параметра, который называется NSFileProtectionComplete. Данный параметр делает так, что файл хранится в зашифрованном виде на диске, и он не может быть доступен, пока устройство находится в заблокированном состоянии. Большинство приложений этот параметр использует. Чтобы преодолеть данную проблему, необходимо отключить автоблокировку экрана. Для этого был разработан автоматизированный способ.

Опытным путем выяснилось, что параметры настроек хранятся в двух специальных файлах типа «plist», которые называются EffectiveUserSettings.plist и PublicEffectiveUserSettings.plist. Их пути в файловой системе iOS соответственно /var/mobile/Library/UserConfigurationProfiles и /var/mobile/Library/UserConfigurationProfiles/PublicInfo. Эти два файла имеют

одинаковые значения. Значения, соответствующие текущему периоду автоблокировки, находятся в списке параметров «restrictedValue». В этом списке находится поле «maxInactivity». Его значение соответствует текущему периоду автоблокировки экрана. Опытным путем, изменяя период автоблокировки в настройках устройства и изучая значение поля, удалось выяснить, что оно может принимать значения 30, 60, 120, 180, 240, 300 и 2147483647, что соответствует периодам 30 сек., 1 мин., 3 мин., 4 мин., 5 мин. и «Никогда» . Пример файла можно увидеть на рисунке 8.

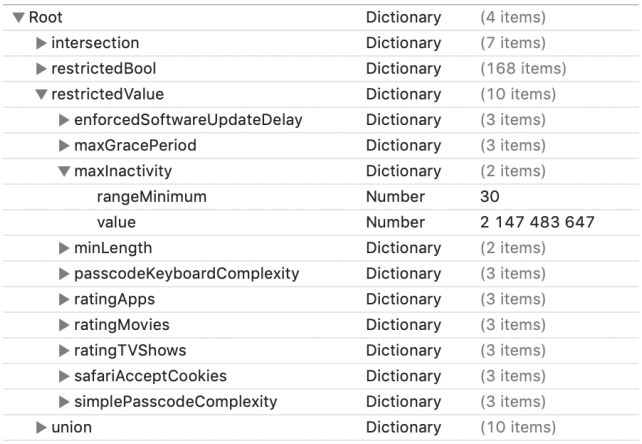

Рис. 8 Содержимое файла EffectiveUserSettings.plist

Поскольку после процедуры Jailbreak телефон имеет права суперпользователя, то можно изменять два файла, в которых хранятся настройки автоблокировки. Однако просто поменять их значения недостаточно, так как телефон автоматически не изменяет настройки отключения экрана. Чтобы изменения вступили в силу, необходимо сделать перезагрузку «SpringBoard» устройства, который представляет собой графический пользовательский интерфейс iOS и управляет графическими службами. Перезагрузку можно выполнить через принудительное завершение процесса «SpringBoard». Изменения сохраняются до перезагрузки устройства.

3.4.2 Реализация агента по отключению автоблокировки

Ввиду особенностей проведения криминалистической экспертизы нельзя устанавливать полноценные приложения на устройство для анализа, поэтому возможно создание только простой утилиты для командной строки, которая впоследствии удаляется.

Основная проблема при этом состоит в том, что официальное средство для разработки iOS приложений Xcode не дает возможности создавать для iOS утилиту командной строки. Для обхода данного ограничения был выбран кроссплатформенный набор инструментов для создания и развертывания программного обеспечения для iOS – theos [16]. Он содержит в себе «iPhoneSDK», который необходим для сборки приложений под ARM[12] архитектуру.

Для написания скрипта агента был выбран язык Objective-C, который используется компанией Apple для написания своих приложений. Суть скрипта заключается в том, что он меняет значение поля «maxInactivity» на 2147483647, что соответствует «Никогда», затем он останавливает процесс «SpringBoard» через вызов команды «killall SpringBoard». В случае, если у пользователя уже отключена автоблокировка, то перезагрузки устройства не произойдет.

При запуске command line tool для версии iOS 11 и выше возникает одна проблема: их запуск невозможен (при запуске выдается результат Killed 9), если приложение не подписано специальным plist-файлом, который содержит тип приложения и его разрешения. Для того чтобы получить файл разрешений из уже существующих программ, использовалась утилита «jtool». С помощью данной утилиты был получен plist-файл с разрешения утилиты «chmod», которая находится в «/usr/bin» на iOS устройстве. Структура файла представлена на рисунке 9.

Кроме того, для подписи приложений необходим специальный сертификат разработчика для Xcode, который можно получить автоматически по имеющемуся аккаунту Apple ID[13]. Затем с помощью встроенной утилиты macOS «cadesign» с парметрами «sha-1» сертифакта и файлом разрешений подписывается приложение агент.

---

[12] ARM – микропроцессорная архитектура с сокращённым набором команд (RISC), разрабатываемая ARM Limited.

[13] Apple ID – учётная запись, которая позволяет пользователям получать доступ к ресурсам Apple.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>platform-application</key>
        <true/>
        <key>com.apple.private.skip-library-validation</key>
        <true/>
        <key>com.apple.private.security.no-container</key>
        <true/>
</dict>
</plist>
```

Рис. 9 Разрешения утилиты «chmod» для iOS

Полная схема взаимодействия прототипа приложения по снятию логического образа реализована следующим образом. По протоколу SSH через утилиту для передачи файлов SCP на устройство загружается агент в папку «/usr/bin». Для того чтобы утилита имела доступ к системе, ей выставляются права через команду «chmod 777 agent», которые дают права на чтение, запись и выполнение для всех групп пользователей. Затем осуществляется вызов агента через команду «agent nolock». После выполнения скрипта агент удаляется через команду «rm agent».

4. Реализация в Belkasoft Evidence Center

После реализации всех компонент прототипа для снятия полного логического образа осуществлялось его внедрение в Belkasoft Evidence Center. В процессе внедрения было обнаружено, что старая архитектура модуля снятия мобильных данных не позволяла добавить новый метод. В связи с чем, было принято решение провести рефакторинг старой модели для добавления возможности простого расширения доступных способов захвата данных с мобильных устройств.

4.1 Рефакторинг старой модели

При анализе старой архитектуры выявились ее существенные недостатки. Опишем их и ее саму более подробно.

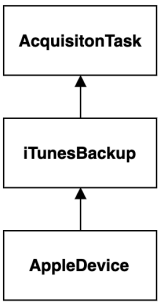

Рис. 10 Упрощенная диаграмма классов старой модели

На рисунке 10 представлена упрощенная диаграмма классов для создания логического образа. Недостатком старой модели являлся слишком большой круг обязанностей класса AppleDevice, а именно, он отвечал за все соединения с устройством и информацию о других подключениях, хранил всю информацию о них, в том числе: UDID[14], имя устройства, версия iOS и т. д. Другим более весомым недостатком было то, что модель не предусматривала создания не только классов для новых типов соединений, но классов для новых способов снятия образов. Все эти факты не позволяли добавить в старую модель логику по установке SSH-соединения, созданию полного логического образа и снятию автоблокировки экрана. Все это стало

[14] UDID – это уникальный идентификатор, который есть у каждого мобильного устройства Apple.

предпосылкой для проведения полного рефакторинга модели, используемой в Belkasoft Evidence Center.

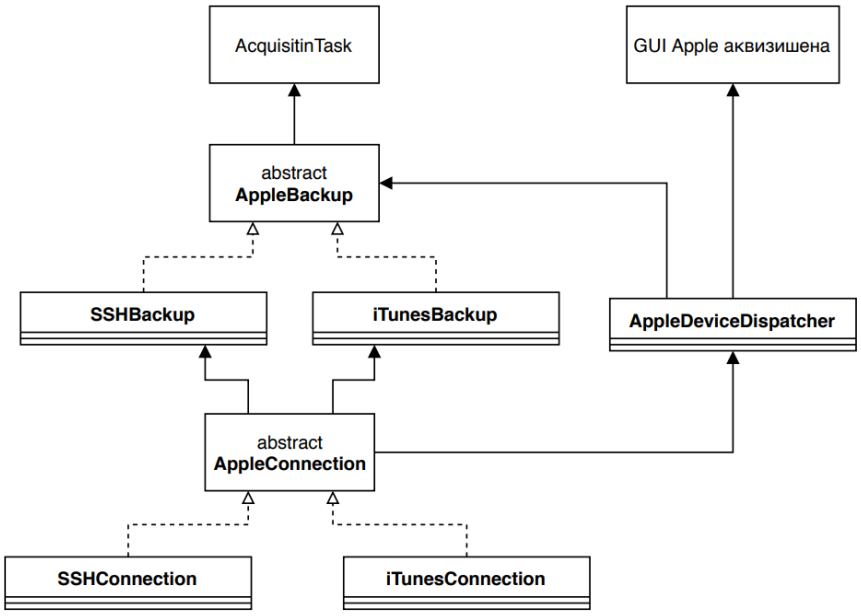

Рис. 11 Диаграмма классов новой модели.

В рамках рефакторинга соединения было принято решение о создании новой модели, которая обеспечивала бы легкость внедрения новых типов соединений. Сначала была проведена работа по разделению класса AppleDevice на новые классы. Вся его логика была разделена на следующие части. Первая часть – это информация об устройстве, реализованная в виде контейнерного класса AppleDeviceInfo, который имеет только поля с данными. Затем была выделено в отдельный класс само соединение. Помимо этого, был создан специальный абстрактный класс AppleConnection, который помогает в создании новых типов соединений. На основе данного абстрактного класса реализован отдельный класс iTunesConnection, в его обязанности входят методы по настройке подключения устройства к компьютеру и получение полной информации об устройстве, которая хранится в AppleDeviceInfo. Следующим – было вынесение контроля за соединениями в новый класс AppleDeviceDispatcher. Он отвечает за получение списка устройств, проверку актуальности списка и контроль за соединением любого типа для каждого устройства, так как класс работает с AppleConnection.

Следующим этапом стало добавление поддержки создания других типов образов, кроме логического. Для этого, в первую очередь, был создан абстрактный класс AppleBackup, имеющий несколько переопределяемых методов для новых классов по созданию образов, который одновременно активно взаимодействует с AppleDeviceDispatcher для постоянного контроля за тем, было ли устройство отключено от компьютера во время создания образа. Логика по созданию резервной копии iTunes была перемещена в производный класс iTunesBackup от AppleBackup. Упрощенная диаграмма классов новой модели представлена на рисунке 11.

Таким образом, рефакторинг старой модели решил все ее основные проблемы и позволил легко внедрить классы для нового подключения и нового способа создания образа.

4.2 Новый графический интерфейс

Для внедрения новой функциональности помимо рефакторинга необходимо было произвести изменения в графическом интерфейсе, отвечающем за создание образов устройств на iOS. Он представлен на рисунке 12. Наиболее значимый недостаток графического интерфейса, применявшегося в Belkasoft Evidence Center, заключался в том, что весь интерфейс был рассчитан на создание только логического образа. Выбор самого устройства, пути сохранения образа и другая дополнительная информация были представлены на одной единственной странице.

Для внедрения новых типов снятия образа интерфейс был переработан следующим образом. Чтобы обеспечить удобство и простоту использования, он был разделен на две страницы, отвечающих за выбор устройств и за выбор метода получения данных. Они представлены на рисунках 13 и 14. В окне выбора метода в соответствии с выбранным типом снятия образа появляется краткое описание метода и кнопки авторизации, необходимые для успешного создания образа. Также присутствует поле с выбором пути для сохранения образа, который помещается в специальный формат компании Belkasoft для хранения данных «belkaml».

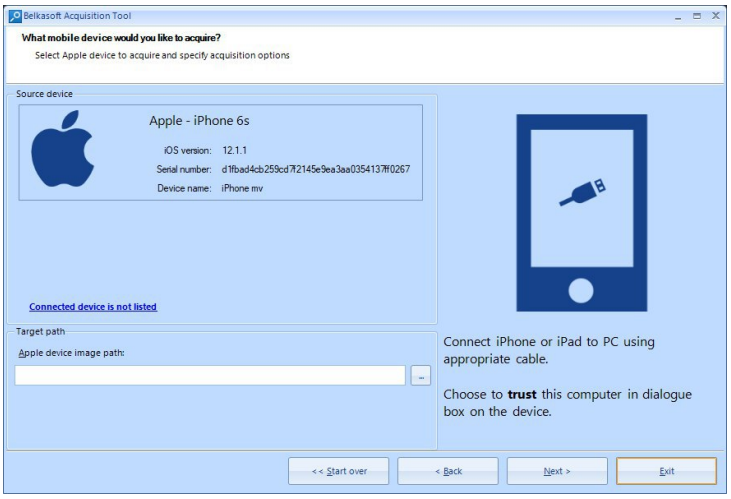

Рис. 12 Старый графический интерфейс снятия образа с iOS устройства.

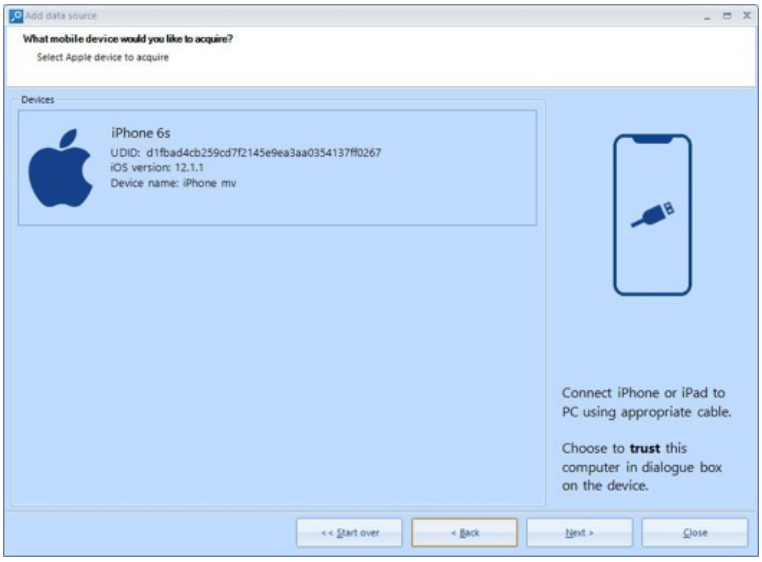

Рис. 13 Окно выбора устройства.

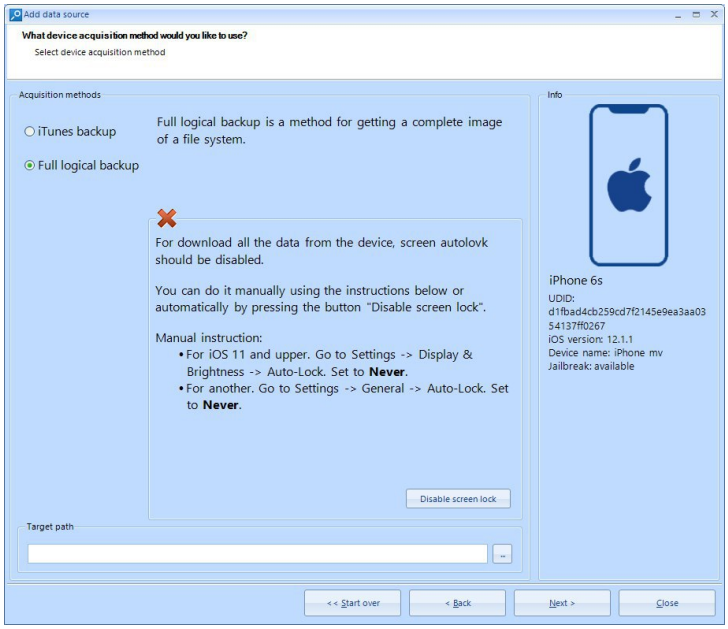

Рис. 14 Окно выбора метода снятия образа с устройства.

4.3 Реализация создания полного логического образа

Разработанный раннее прототип после проведения рефакторинга и соответствующих изменений в графическом интерфейсе был интегрирован в Belkasoft Evidence Center.

Создание образа возможно после проведения процедуры авторизации Jailbreak на устройстве – это реализовано через нажатие специальной кнопки «verify jailbreak». После проверки статус Jailbreak отображается справа в блоке информации об устройстве. Это сделано в целях исключения возникновения проблем при создании образов на устройствах без проведенной процедуры Jailbreak, так как на них это невозможно.

Снятие автоблокировки экрана происходит через нажатие кнопки «Disable screen lock». Статус отображается таким же образом, как статус Jailbreak. Однако данное действие является необязательным, пользователь может самостоятельно ее отключить, для чего была подготовлена и размещена рядом с кнопкой отключения соответствующая инструкция. Вид данной страницы отображен на рисунке 14.

После всех выполненных ранее действий, нажатием на кнопку «Next» начинается процесс по созданию безопасного SSH-соединения и само снятие полного логического образа. Во время снятия отображается скорость загрузки в мегабайтах в секунду и размер загруженной информации в гигабайтах, что отображено на рисунке 15.

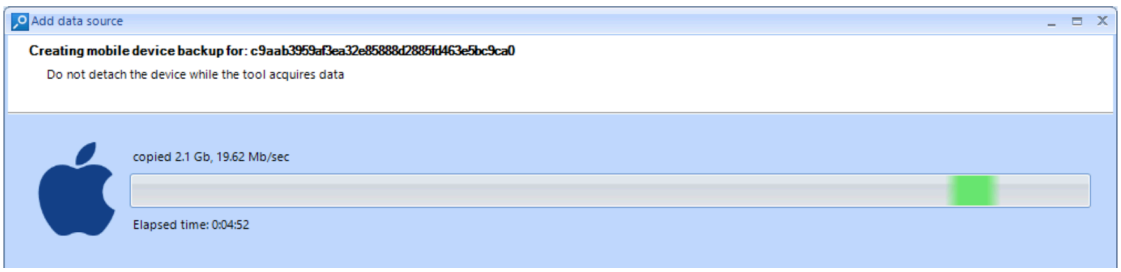

Рис. 15 Информация по снятию образа

5. Апробация

5.1 Описание устройств

Разработка проекта и дальнейшая его апробация осуществлялись с использованием устройств, представленных в таблице 2.

Устройство	Версия iOS	Jailbreak	Архитектура процессора
iPhone 4S	9.3.5	Phoenix	Armv7
iPhone 6	11.1.3	Electra	Arm64
iPhone 6S	12.1.4	Unc0ver	Arm64

Таблица 2 Список устройств.

Список устройств был выбран с учетом того, чтобы охватить различные версии iOS и различные архитектуры процессора.

5.2 Сравнение времени снятия образа на описанных устройствах

Для сравнительного анализа времени снятия образа использовались устройства, представленные в пункте 5.1. Все они были подключены к компьютеру через USB 3.0. Образы создавались с помощью Belkasoft Evidence Center. Для замера времени использовался специальный класс языка C# Stopwatch, который представляет методы для точного измерения затраченного времени на выполнение выбранной части кода. Для анализа использовался временной интервал между первым получением данных от «pipe» вызова «ssh.exe» и моментом, когда данные перестают приходить. Результаты исследования представлены в таблице 3.

Устройство	Время (мин.)	Размер (Мб)	Средняя скорость загрузки (Мб/с)
iPhone 4S	13	3920	5
iPhone 6	18,6	11200	10
iPhone 6S	6,5	10800	27,6

Таблица 3 Сравнительный анализ времени создания образа на разных устройствах.

Помимо этого было проведено снятие образов с помощью продукта Elcomsoft iOS Forensic Toolkit. Все временные показатели, размер и средняя

скорость загрузки образа оказались такими же, как и у Belkasoft Evidence Center.

5.3 Сравнение логического и полного логического образов

Для проведения сравнительного анализа были подготовлены тестовые данные на устройстве iPhone 6S. Были установлены следующие программы для обмена сообщениями: Telegram, WhatsApp, Facebook и WeChat. Для них были созданы тестовые аккаунты. В каждом из приложений организована переписка с другим тестовым аккаунтом.

Далее было подготовлено два образа. Первый из них – логический образ, второй – полный логический. Оба образа были созданы с помощью программы Belkasoft Evidence Center. Затем был проведен сравнительный анализ двух образов, результаты анализа которого представлены в таблице 4. Критерием оценивания послужило количество артефактов в каждом из приложений.

Приложение	**Логический образ**	**Полный логический образ**
Facebook	0	45
Telegram	0	167
WeChat	0	80
WhatsApp	10	10
Safari	173	6266

Таблица 4 Сравнительный анализ образов

Как можно видеть из таблицы 4, полный образ содержит намного больше информации. Примером служит приложение Telegram, которое в логическом образе отсутствует, однако в полном образе из него извлеклись все открытые чаты. Фрагмент сообщений на устройстве и в полученном образ представлены на рисунках 5 и 6.

Рис. 5 Фрагмент чата приложения Telegram на устройстве

☐		373499720	637821205		10/19/2018 2:41:4...	Come on people
☐		373499720	637821205		10/19/2018 2:41:4...	Jake is better
☐		637821205	373499720		10/19/2018 2:41:2...	You are the best lllllol
☐		637821205	373499720		10/19/2018 2:41:0...	Hello

Рис. 6 Извлеченный фрагмент чата в Belkasoft Evidence Center

5.4 Сравнение образа до и после отключения автоблокировки экрана

Для сравнительного анализа использовался iPhone 6S, характеристики которого описаны выше. Было создано два образа. Первый образ создавался при включенной автоблокировке экрана, а второй – при отключенной. После создания образов стало ясно, что количество данных будет отличаться, поскольку в первом случае размер образа составил 8025028 килобайт, а во втором – 9902550 килобайт. Разница между размерами образов составила 1877522 килобайта. Помимо этого, критерием для анализа послужило, как и в случае сравнения логического и полного логического образов, количество артефактов. Результаты анализа представлены в таблице 5.

	автоблокировка	**автоблокировка отключена**
Подключения Bluetooth	0	116
Изображения	6548	8020
Документы	620	774
Telegram	346	346

Таблица 5 Сравнительный анализ образов.

Таким образом, полученные результаты исследования подтверждают вывод о том, что при включенной автоблокировке экрана устройства теряется достаточное количество данных.

Заключение

В рамках данной работы были выполнены следующие задачи:

- исследованы способы получения данных из файловой системы;
- исследована суть уязвимости «tfp0» и возможность применения данной уязвимости для получения доступа к данным на мобильном устройстве
- реализовано приложение для взаимодействия между устройством и персональным компьютером;
- реализовано приложение-прототип для снятия образа и внедрено в Belkasoft Evidence Center;
- проведена апробация.

Список литературы

[1] Н. Н. Федотов. Форензика - компьютерная криминалистика. – URL: http://www.bnti.ru/showart.asp?aid=998&lvl=01.02.01 (дата обращения 14.10.2018).

[2] Sarah Edwards. The iOS of Sauron: How iOS Tracks Everything You Do. – URL: https://files.sans.org/summit/Digital_Forensics_and_Incident_Response_Summit_2016/PDFs/iOS-of-Sauron-How-iOS-Tracks-Everything-You-Do-Sarah-Edwards.pdf (online; accessed: 14.10.2018).

[3] Belkasoft. Belkasoft Evidence Center. – URL: https://belkasoft.com/ec (online; accessed: 20.02.2019).

[4] Gordon Kelly. Apple iOS 12.2 Release: Should You Upgrade? – URL: https://www.forbes.com/sites/gordonkelly/2019/03/26/apple-ios-12-2-release-should-you-upgrade/#7d57365c42ea (online; accessed: 25.05.2019).

[5] Kirill Domrin. What is jailbreak?. – URL: https://iphone-gps.ru/jailbreak/chto-takoe-jailbreak (online; accessed: 20.02.2019).

[6] Todd Shields, Adam Satariano. 'Jailbreaking' of IPhones to Add Apps Backed by U.S. – URL: https://www.bloomberg.com/news/articles/2010-07-26/apple-iphone-users-have-u-s-blessing-to-jailbreak-add-own-applications (online; accessed: 15.12.2018).

[7] Jailbreak. – URL: https://www.theiphonewiki.com/wiki/Jailbreak (online; accessed: 10.12.2018).

[8] SSH PROTOCOL. – URL: https://www.ssh.com/ssh/protocol/ (online; accessed: 5.03.2019).

[9] Tatu Ylonen. The Secure Shell (SSH) Protocol Architecture. – URL: https://rfc2.ru/4251.rfc/original (online; accessed: 10.03.2019).

[10] Elcomsoft iOS Forensic Toolkit. – URL: https://www.elcomsoft.ru/eift.html (online; accessed: 25.05.2018).

[11] Oxygen Forensic Extractor. – URL: https://www.oxygen-forensic.com/en/products/oxygen-forensic-extractor (online; accessed: 10.03.2019).

[12] Brandon Azad. Voucher_swap: Exploiting MIG reference counting in iOS 12. – URL: https://googleprojectzero.blogspot.com/2019/01/ (online; accessed: 25.04.2019).

[13] Apple Support. About macOS, iOS, and iTunes server host connections and iTunes background processes. – URL: https://support.apple.com/en-gb/HT201999 (online; accessed: 22.03.2019).

[14] Apple Support. TCP and UDP ports used by Apple software products. – URL: https://support.apple.com/en-gb/HT202944 (online; accessed: 25.05.2018).

[15] File with .plist extension. – URL: https://open-file.ru/types/plist (online; accessed: 19.04.2019).

[16] Theos homme page. – URL: https://github.com/theos/theos/wiki. (online; accessed: 19.04.2019).