Санкт-Петербургский государственный университет

Белошапкин Михаил Юрьевич

Выпускная квалификационная работа

Извлечение данных keychain из облачного хранилища iCloud

Уровень образования: бакалавриат

Направление *02.03.03 «Математическое обеспечение и администрирование информационных систем»*

Основная образовательная программа *СВ.5006.2019 «Математическое обеспечение и администрирование информационных систем»*

Научный руководитель:
к.т.н., доцент кафедры системного программирования Ю.В. Литвинов

Консультант:
специалист по вирусной безопасности АО «Лаборатория Касперского»
М.В. Виноградов

Рецензент:
руководитель отдела разработки ПО ООО «Белкасофт» Н. М. Тимофеев

Санкт-Петербург
2023

Saint Petersburg State University

Beloshapkin Mikhail

Bachelor's Thesis

Extraction of keychain data from iCloud cloud storage

Education level: bachelor

Speciality *02.03.03 "Software and Administration of Information Systems"*

Programme *CB.5006.2019 "Software and Administration of Information Systems"*

Scientific supervisor:
Software Engineering Chair, C.Sc Y.V. Litvinov

Consultant:
Digital forensic specialist at «Kaspersky Lab»
M.V. Vinogradov

Reviewer:
Head of software development at «Belkasoft» N.M. Timofeev

Saint Petersburg
2023

Оглавление

1. Введение

Современная криминалистика имеет ряд отдельных направлений, одно из которых – цифровая криминалистика. Данная область специализируется на извлечении и последующем анализе данных с персональных компьютеров, мобильных устройств, облачных хранилищ. Полученная таким образом информация может сыграть важную в роль в ходе проведения криминалистических экспертиз, и также может быть использована для предотвращения преступлений.

Особую важность для экспертов в области цифровой криминалистики представляют набирающие в последние годы популярность облачные сервисы. Они позволяют пользователям безопасно загружать и хранить различные данные на удаленных серверах, а некоторые сервисы также могут автоматически синхронизировать данные на авторизованных устройствах. В качестве примера можно привести встроенный в операционную систему iOS облачный сервис iCloud. Помимо хранения фотографий, видео- и аудиофайлов, заметок, контактов, резервных копий, в iCloud также реализован iCloud keychain – приложение, которое позволяет загружать пароли от различных аккаунтов, пароли от WiFi-сетей, а также номера банковских карт и другую конфиденциальную информацию, таким образом избавляя пользователя от необходимости запоминать пароли и вводить их вручную. Очевидно, что эти данные представляют особую ценность для их владельца, а, следовательно, и для специалистов, которые занимаются исследованиями в области цифровой криминалистики. Таким образом, возникает необходимость в наличии инструмента, который позволяет для конкретной учетной записи iCloud автоматически извлекать данные iCloud keychain при наличии возможности авторизоваться. Такой подход позволяет представить полученную информацию в удобном для криминалиста виде, а также извлечь те данные, к которым нет доступа в стандартном пользовательском интерфейсе.

На рынке программного обеспечения для цифровой криминалистики существует программный пакет Belkasoft X, разработанный компа-

нией Belkasoft, в функциональность которого входит автоматическое извлечение данных из облачных хранилищ. В нем есть модуль, который позволяет после авторизации в iCloud автоматически выгружать фотографии, данные электронной почты, заметки и даже резервные копии для устройств на операционной системе iOS, но на данный момент в этом продукте не реализовано извлечение данных iCloud keychain. Поэтому было принято решение реализовать инструмент для скачивания iCloud keychain с целью его последующего интегрирования в продукт Belkasoft X.

2. Постановка задачи

Целью данной работы является реализация инструмента для извлечения данных keychain из облачного хранилища iCloud в рамках продукта Belkasoft X. Для достижения этой цели были сформулированы следующие промежуточные задачи.

1. Сделать обзор существующих решений.
2. Изучить и применить методы исследования клиент-серверных приложений.
3. Составить схему общения встроенного системного приложения с серверами Apple.
4. Изучить схему дешифровки данных.
5. Реализовать инструмент для извлечения данных keychain из облачного сервиса iCloud.
6. Провести тестирование и апробацию реализованного модуля.

3. Обзор

3.1. Существующие решения

На рынке программного обеспечения для цифровой криминалистики существует только одно решение, которое позволяет извлекать данные keychain из облачного хранилища iCloud:

Elcomsoft phone breaker – программный комплекс, предоставляющий возможность извлечения различных данных iCloud, в том числе keychain. Код проекта является закрытым, более того, провести реверс-инжиниринг данного инструмента не представляется возможным из-за отсутствия его в свободном доступе, поэтому у нас есть возможность ознакомиться лишь с его функциональностью. Данную информацию можно найти на официальном сайте, а также из выложенных в открытый доступ презентаций. Для получения доступа к iCloud keychain требуется ввести следующие данные: Apple Id, пароль, код двухфакторной аутентификации, сгенерированный одним из авторизованных устройств, код блокировки или пароль от одного из авторизованных устройств, после чего происходит скачивание и дешифровка данных.

3.2. Литература

На данный момент существует несколько статей, описывающих процесс исследования iCloud keychain. Приведем некоторые из них:

«Извлечение резервной копии данных устройства на операционной системе iOS из облачного хранилища iCloud» [8], выпускная квалификационная работа М. В. Виноградова. В ней описывается процесс исследования клиентов iCloud для различных операционных систем. В качестве результата автором получены все возможные схемы авторизации, а также алгоритм загрузки резервных копий из облачного хранилища iCloud. Также в данной работе подробно рассказывается о специальных инструментах и их применении в реальном исследовании.

In the depth of iCloud keychain [3]. В данной статье описывается

способ перехвата трафика системных приложений iOS. Автор рассказывает о том, какие алгоритмы применяются для шифрования данных, а также о принципе работы протокола SRP-6a и его роли процессе авторизации. Однако на первом этапе исследования наибольший интерес для нас представляет описание внутреннего API для загрузки или синхронизации данных keychain. Проблема состоит в том, что информация, приведенная в статье является устаревшей, а, следовательно, API для доступа к данным мог неоднократно поменяться. Тем не менее, эта статья дает примерное представление о том, как именно просходит общение клиента и сервера.

Apple iCloud inside out [9]. Данный документ содержит уже более подробное описание API, а также схем извлечения данных из iCloud. Информация приведена не только для разных типов аккаунтов (с или без двухфакторной авторизации), но и для разных версий iOS. Также стоит отметить, что в этом документе автор сообщает не только о том, какие API используются при скачивании записей keychain, но и о том, как правильно сформировать нужные запросы и какие данные необходимо в них отправить. Эта информация представляет наибольшую ценность, так как позволяет понять, на что именно стоит обратить внимание при дальнейшем анализе сетевого трафика.

В качестве еще одного примера можно привести презентацию компании Elcomsoft «Breaking into the iCloud keychain» [1]. В ней представлены схемы авторизации для обычной и двухфакторной аутентификации, также описан принцип работы протокола SRP, последовательность получения токенов для скачивания и дешифровки данных keychain. Данная презентация также является устаревшей, тем не менее, ее изучение помогло в общих чертах представить стратегию дальнейшего исследования.

3.3. Инструменты

Стоит кратко описать набор инструментов, с помощью которых можно проводить исследования клиент-серверных программ:

SSL Killswitch – утилита, котрая позволяет отключить проверку SSL-сертификата в программах на операционных системах iOS и macOS. Данный инструмент может пригодиться тогда, когда необходимо получить дамп сетевого трафика для различных приложений. SSL-killswitch вносит изменения в логику работы функций, которые отвечают за обработку SSL/TLS-соединений, таким образом отключая проверку сертификата системы. Данную утилиту можно установить при помощи Cydia[1] на устройство с прошивкой Jailbreak[2].

Burp Suite – представляет собой платформу для тестирования прикладных клиент-серверных приложений. Этот инструмент позволяет сконфигурировать отладочный прокси-сервер, то есть данный инструмент также можно использовать для перехвата сетевого трафика с целью дальнейшего его исследования. Для того, чтобы получить дамп трафика, на телефоне необходимо выставить ip-адрес и порт прокси-сервера, а также подтвердить его SSL-сертификат. При этом компьютер, на котором запущен Burp Suite, и телефон, сетевой трафик которого прослушивается, должны быть в одной подсети.

Frida – инструмент, который позволяет внедрять пользовательские скрипты в уже запущенные программы с целью их исследования. Поддерживает операционные системы Windows, macOS, GNU/Linux, iOS, watchOS, tvOS, Android, FreeBSD, и QNX. С помощью этого инструмента можно перехватывать вызовы функций, а также подменять функции, таким образом внося изменения в логику работы программы. Таким образом, появляется возможность анализа логики программы в реальном времени.

Wireshark – программа, которая представляет собой отладчик трафика, в том числе для компьютерных сетей Ethernet. Как инструмент для исследования, он имеет одно существенное преимущество: наличие API в виде библиотеки Pyshark, которая позволяет автоматизировать обрабаботку данных из снимка трафика.

[1]Cydia– приложение для операционной системы iOS, позволяющее устанавливать сторонние программные пакеты.

[2]Jailbreak – специальная прошивка, с помощью которой можно устанавливать сторонние программы.

HttpToolkit – инструмент для перехвата HTTP/HTTPS трафика. В качестве основного преимущества данной программы можно отметить простоту ее использования: например, для перехвата трафика локально установленного браузера требуется только выбрать соответствующий пункт меню в пользовательском интерфейсе, не производя при этом никаких дополнительных настроек.

4. Исследование

4.1. Перехват трафика с помощью Burp Suite

Для снятия и последующего анализа сетевого трафика использовалось устройство на операционной системе iOS версии 14.8. Снятие трафика осуществлялось при помощи MITM-атаки, для этого был применен инструмент Burp Suite, с помощью которого был сконфигурирован отладочный прокси-сервер, в настройках телефона был выставлен его ip адрес и порт. Чтобы прокси-сервер мог установить с клиентом TLS-соединение и дешифровать идущий через него трафик, на телефоне необходимо было также подтвердить SSL-сертификат прокси-сервера. Атакуемая системная программа обменивается трафиком с прокси-сервером, при этом Burp Suite расшифровывает трафик своим сертификатом, а зашифровывает при помощи оригинального сертификата настоящего сервера, к которому непосредственно обращается приложение. Однако при данном подходе есть одна существенная проблема, а именно используемый в системных программах iOS SSL-pinning. Суть SSL-pinning заключается в том, что SSL-сертификат сервера, на который отсылаются запросы, хранится непосредственно внутри приложения, игнорируя при этом хранилище сертификатов устройства, а значит подтвержденный сертификат прокси-сервера не будет валидным. Для решения этой проблемы была установлена утилита SSL-killswitch. Однако, как оказалось, она работает не со всеми системными приложениями iOS, поэтому в данном случае такая схема работает только с отключенной дешифровкой трафика на доменное имя https://setup.icloud.com, а для исследования необходимо получить полный дамп трафика, чтобы была возможность корректно составить схему общения клиента и сервера.

4.2. Перехват трафика с помощью Frida

Для получения полного дампа трафика использовался метод, описанный в статье «Capturing and Decrypting HTTPS Traffic From iOS

Apps Using Frida». Для реализации данного метода необходимо иметь компьютер на операционной системе Mac OS, а также устройство с root-правами. При запуске Frida в аргументах командной строки указываются pid процесса, в который мы хотим внедрить скрипт, файл, куда будут выводиться TLS ключи, а также путь к самому скрипту. С помощью данного скрипта находятся подключаемые библиотеки, выбирается «libboringssl.dylib», которая в свою очередь содержит функцию SSL_CTX_set_info_callback, которая используется для получения информации об объектах SSL. Затем в key_log_callback записывается указатель на содержащуюся в этом же скрипте функцию-логгер, которая и выводит данные о TLS-ключах в указанный файл. Перехват непосредственно самого трафика с устройства осуществляется с помощью встроенной в MacOS утилиты rvictl (Remote Virtual Interface Tool). Она создает интерфейс, который затем можно использовать для перехвата трафика инструментом tcpdump. В итоге у нас должны получиться несколько файлов с TLS-ключами (количество файлов равно число процессов, в которые мы внедряем скрипт), а также файл с расширением pcap с зашифрованным трафиком. Для расшифровки трафика TLS-ключами в нашем случае используется инструмент Wireshark.

На следующем этапе необходимо найти те процессы, которые отвечают за общение с сервером во время авторизации и скачивания данных keychain. Для этого был изучен список процессов, после чего был получен список для снятия:

- akd;
- cloudd;
- preferences;
- sbd.

Таким образом, внедряя скрипт в каждый из этих процессов по время авторизации и скачивания, удалось получить полный дамп трафика. Далее можно приступить к его анализу.

4.3. Веб-версия iCloud

Стоит отметить, что в браузерной версии клиента iCloud пользователь имеет доступ к очень ограниченному множеству данных, в частности, в веб-версии отсутствует доступ к данным Keychain. Однако исследования различных типов клиентов могут быть полезными в случае, когда у программы нет публичного API, так как это позволяет составить более подробное представление о том, за что отвечают те или иные запросы. Также такой подход имеет одно существенное преимущество: в веб-приложениях у нас есть доступ к исходному коду клиента. Для исследования трафика в веб-версии использовался инструмент HttpToolkit. Удалось выяснить, что веб-версия использует несколько иную схему авторизации, которая не предусматривает получение токенов для keychain, а значит интересующих нас данных там, естественно, не оказалось.

4.4. iOS 14.8

Перед тем, как перейти к дешифровке данных, нужно непосредственно их извлечь. Все последующие эксперименты проводились для устройства iPhone 7 с установленной на нем iOS 14.8. На первом этапе исследования сетевого трафика мы отталкивались от предположения, что в за загрузку данных keychain отвечает сервис gateway, так как после прохождения авторизации на него отправлялось большинство запросов. Найти подробное описание API в открытых источниках для gateway не удалось, более того, этот сервис принимает и отправляет данные в формате protobuf, то есть эти сообщения необходимо сначала декомпилировать. Поэтому для того, чтобы данные запросов предстали в явном виде, был написан скрипт на языке программирования Python с использованием библиотеки pyshark. Принцип работы данного скрипта заключается в следующем: он перебирает все пакеты из дампа трафика, и, если их содержимое имеет формат protobuf, происходит парсинг и декомпиляция сообщений, которые затем сохраняются в специальный файл. Стоит уточнить, что структура protobuf-сообщений определяется

так называемыми классами, которые необходимо передавать компилятору для преобразования сообщения в байтовый массив, а также для декомпиляции. В данном случае использовалась утилита protoc, которая позволяет восстанавливать сообщения без указания соответствующего класса, но при таком подходе теряется часть информации (например, имена полей). Так был получен полный список запросов и ответов. В ходе анализа этих данных наибольшее внимание привлек ответ на запрос, который был осуществен на путь /sync (это же API используется при синхронизации данных, как было получено в одном из экспериментов). Этот ответ содержал список из зашифрованных данных, а также ключи шифрования. В связи с этим была выдвинута гипотеза о том, что именно это API используется для извлечения данных keychain. Чтобы в этом убедиться, был проведен следующий эксперимент: брались два iOS устройства, сетевой трафик одного из них прослушивался с помощью Frida (скрипт внедрялся в демон cloudd, именно он отвечает за отправку запросов на gateway), на втором устройстве происходило создание новой записи keychain, после чего данные сразу синхронизировались, то есть только что созданная запись выгружалась и на второе устройство. По итогу этого эксперимента выяснилось, что именно на /sync приходит соответствующее protobuf-сообщение, которое содержит блок закодированных данных, поле с названием wrapedKey и несколько различных идентификаторов. То есть данное API вероятнее всего отвечает за скачивание интересующих нас данных. Проблема состоит в том, что сообщения с запросами на получение этих данных имеют довольно сложную структуру. Более того, в материалах Elcomsoft содержится информация о том, что данные keychain могут скачиваться в двух режимах: в режиме Sync (то есть через использование сервиса gateway) и в режиме Recovery (с помощью keyvalue-service). Поэтому, прежде чем приступать к более подробному анализу, было принято решение провести еще ряд экспериментов.

Так было выяснено, что после полного сброса устройства в извлечении данных участвует сервис keyvalue-service. Сначала клиент обращается к /setASPToken, после этого производится запрос на /sync. В ответ

на последний запрос приходят данные в формате plist, которые представляют собой словарь. В данном словаре ключ указывает на то, какие данные там хранятся (например, ключ com.apple.icdp.backup.WiFi-tomb означает, что текущая запись содержит данные о Wifi паролях, com.apple.icdp.backup.Passwords-tomb указывает на то, что запись хранит пароли для веб-сайтов и приложений). Значениями данного словаря являются так называемые Tomb-записи. Каждая такая запись представляет собой закодированный в plist блок данных, который содержит поля backup и keybag. Данные поля представляют собой закодированные в формате ASN.1[3] списки, причем их структура очень похожа на ту, что была описана в презентации Elcomsoft. Раскодированный backup содержит список из зашифрованных записей keychain, раскодированный Keybag хранит различные ключи шифрования, его содержимое представлено на Листинге 1:

Листинг 1: Раскодированные данные Keybag

```
SEQUENCE {
 OCTETSTRING 5645525300000004000000055459504...
 SEQUENCE {
    SEQUENCE {
       SET {
          SEQUENCE { ... }
          SEQUENCE {
             UTF8String 'DeviceGestalt'
             SET {
                SEQUENCE {
                   UTF8String 'ModelName'
                   UTF8String 'iPhone'
                ...
       }
       OCTETSTRING 3045022061EE7CE9...
    }
 }
 SET {
    SEQUENCE {
       UTF8String 'ywB6hhEwr9UtFbfasSvwoiDiHn'
       OCTETSTRING 0100D67E71378D47F9CB6A3B2922EEB8E060...
    }}}
```

[3]Introduction to ASN.1, URL: https://www.itu.int/en/ITU-T/asn1/Pages/introduction.aspx (Date: 04.04.2023).

В раскодированной последовательности нас интересуют первый и последний блоки данных. Первый элемент этой последовательности преставляет собой байтовый массив, который мы назовем BackupKeybag, последний блок данных – это пара, состоящая их хэша авторизованного в учетной записи устройства и еще одного байтового массива, который мы назовем KeybagWrapedKey, то есть каждому множеству записей Keychain соответствует единственное устройство.

Стоит уточнить, что BackupKeybag представляет собой не что иное как закодированное множество ключей. Более того, байтовый массив с аналогичной структурой используется для хранения ключей для других сервисов (например, для iTunes или резервных копий). В этом массиве данные сгруппированы следующим образом: первые четыре байта кодируют тэг (ASCII строка), следующие четыре байта хранят длину данных, после этого идет ключ. Данная структура подержит в себе ключи, которые затем будут использованы для дешифровки данных keychain.

4.5. Escrow-proxy

В официальном документе «Apple Platform Security» [4], а также в статье «In the Depth of iCloud Keychain» [3] упоминается тот факт, что для извлечения данных keychain используются два сервиса – keyvalue и escrow-proxy. В ходе анализа трафика было выяснено, что на escrow-proxy отправляются всего три запроса, в каждом из которых данные передаются в формате plist:

- /get_records – осуществляет получение специальных записей для каждого авторизованного в учетной записи устройства, каждая из этих записей содержит различную информацию (например, количество оставшихся попыток авторизации).

- /srp_init – в теле запроса передается хэш устройства, для которого мы хотим получить BackupBagPassword, а также специальный ключ, который формируется на основе кода разблокировки этого устройства.

- /recover – в запросе передается хэш устройства, в ответ приходит словарь, в котором и содержится BackupBagPassword для конкретного устройства.

Следует отметить, именно последний запрос представляет наибольший интерес, так как в ответ на него приходит plist-словарь, в котором содержится поле BackupBagPassword, который представляет собой некоторое guid-значение, оно является уникальным для каждого устройства.

По итогу была получена модель взаимодействия клиента и сервера, описанная на рисунке 1:

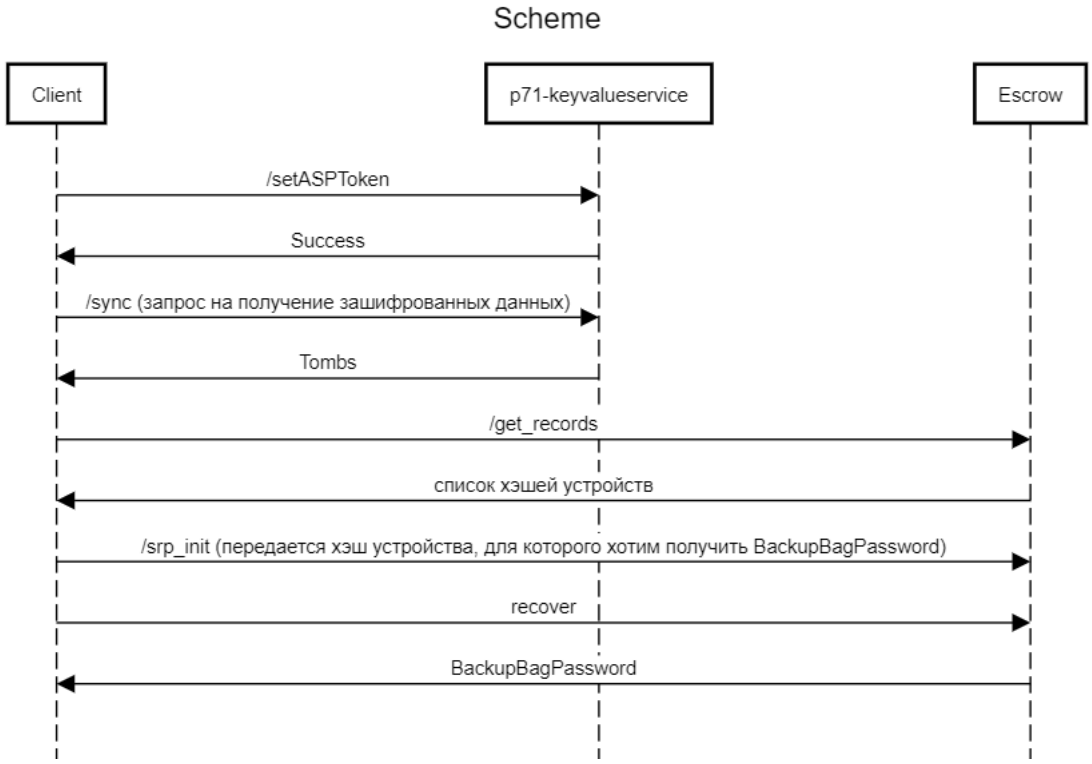

Рис. 1: Схема взаимодействия клиента и сервера

4.6. Получение ключа шифрования для Keychain

После получения записей keychain с сервера их нужно дешифровать. В процессе дешифровки данных участвуют два сервиса – securityd и sbd. На этом шаге стоит отметить, что исходный код некоторых системных программ для операционных систем MacOS и iOS можно найти в открытом доступе. Так, на GitHub выложен частичный исходный код securityd[4]. В ходе анализа было найдено место, где происходит дешифровка записей. Выяснилось, что для осуществления расшифровки необходимо сначала получить ключ размером 32 байта. Для генерации данного ключа Securityd использует криптографические примитивы из библиотеки corecrypto. Извлечение данного ключа происходит по описанной ниже схеме[5]:

1. Из escrow proxy извлекается BackupBagPassword – ключ, который используется для дешифровки данных в keybag.

2. Из полученного ключа генерируется новый ключ с помощью алгоритма PBKDF2, при этом в качестве аргументов выступают следующие значения.

   (a) Соль: массив, который состоит из единственного нулевого байта.

   (b) Число итераций: 0x14.

   (c) Выходной размер: 0x400.

3. К полученному на прошлом шаге ключу применяем функцию ccec_generate_key_determenistic , таким образом генерируем новый ключ.

4. К байтовому массиву keybagWrapedKey, который мы получаем из keybag, а также к ключу из пункта (3) применяется алгоритм RFC6637 и получаем unwrapedKey.

[4]https://github.com/apple-opensource/Security

[5]реализации функций pbkdf2_sha256, pbkdf2_sha1, ccec_generate_key_determenistic, ccec_rfc6637_unwrap_key были взяты из библиотеки corecrypto.

5. Далее применяем алгоритм PBKDF2 с функцией хэширования SHA1. Параметры этого алгоритма, а именно соль и количество итераций, извлекается из BackupKeybag. На данном шаге мы получили ключ размером 32 байта, который затем будет использован для расшифровки данных Keychain.

Общая схема данного алгоритма изображена на рисунке 2:

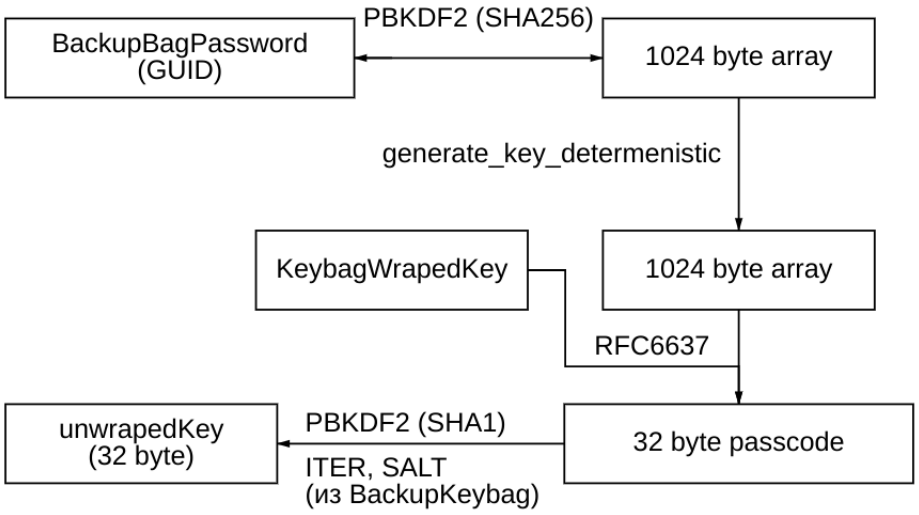

Рис. 2: схема получения unwraped key

Проблема заключается в том, что на GitHub выложен не весь исходный код сервиса Securityd, как следствие, часть логики, отвечающей за дешифровку, отсутствует.

4.7. iPhoneDataprotection

iPhoneDataprotection[6] – это инструмент с открытым исходным кодом, в котором реализована логика дешифровки данных из резервных копий iOS. Также в нем предусмотрена возможность дешифровки данных keychain. При анализе исходного кода данного инструмента была найдена функция, генерирующая 32 байтовый ключ шифрования для записей keychain, причем логика ее работы отличалась от той, что описана на Рис. 3. Однако при этом все остальные шаги в данном инструменте были реализованы. Тогда возникло предположение, что ес-

[6]https://github.com/moloch–/iPhoneDataprotection

ли каким-либо образом извлечь ключ шифрования для iCloud keychain, то остальную часть iPhoneDataprotection можно использовать для дешифровки записей. Для проверки этой гипотезы был написан скрипт, который перехватывает вызов функции ccpbkdf2_hmac (она соответствует последнему шагу генерации ключа на Рис. 2) и сохраняет сгенерированный ключ в специальный файл. С помощью Frida данный скрипт был внедрен в Securityd, после чего была осуществлена авторизация и скачивание данных из iCloud. Полученный ключ был подставлен вместо вызова соответствующей функции, также предварительно были сохранены ответ сервера, содержащий Tomb-записи, и извлечен BackupBagPassword для соответствующего устройства. На этих исходных данных были запущены скрипты из iPhoneDataprotection, после чего данные keychain успешно расшифровались. Таким образом была получена вся логика дешифровки данных.

4.8. Структура BackupKeybag

BackupKeybag – это байтовый массив, в котором хранится различная информация. Стоит более подробно остановиться на описании его содержимого. Данный массив состоит из блоков, каждый блок имеет следующую структуру:

1. Тэг (4 байта) – закодированная ASCII строка
2. Длина (4 байта) – длина последующего блока данных
3. Сами данные

В свою очередь эти блоки (тэг, длина, данные) объединяются в структуры, каждая следующая структура отделяется от предыдущей блоком, который содержит тэг «UUID». При этом данные с различными тэгами задают различные параметры шифрования. Самая первая такая структура содержит в себе следующие тэги:

- Блок с тэгом «ITER» задает количество итераций алгоритма PBKDF2 в последнем шаге схемы получения 32 байтового ключа разблокировки.

- Блок с тэгом «SALT» задает соль для того же алгоритма.

Последующие структуры содержат в себе следующие важные для нас блоки:

- Блок с тэгом «CLAS» кодирует «класс» записей. Класс определяет конфигурацию, в которой эти данные доступны (например, только после разблокировки устройства).

- Блок с тэгом «WPKY» определяет ключ для данного класса, который затем будет использован на последнем шаге дешифровки данных.

4.9. Разблокировка BackupKeybag

После того, как структура BackupKeybag декодирована, можно получить ключ шифрования для каждого класса, они генерируются с помощью алгоритма AES, при этом ключом является unwrapedKey, полученный в пункте 5.4, в качестве данных выступает ключ в тэгом «WPKY» (Рис. 3).

4.10. Дешифровка Keychain

Как уже было упомянуто ранее, зашифрованные данные keychain мы можем получить при раскодировании поля backup в Tomb-записях. Таким образом мы получим список из байтовых массивов. В ходе исследования выяснилось, что эти данные имеют ту же структуру, что и данные из резервных копий. Каждый массив кодирует класс записей, ключ шифрования, а также сами данные keychain.

После того, как BackupKeybag был разблокирован, мы получили новый ключ шифрования для каждого класса записей (первый шаг на

Рис. 3). Далее с помощью алгоритма AES мы получаем новый ключ, при этом в качестве данных выступает wrapedKey, который закодирован в байтовом массиве keychain. После этого новый сгенерированный ключ используется для дешифровки данных keychain алгоритмом AES.GCM. На выходе получается закодированная в ASN.1 последовательность.

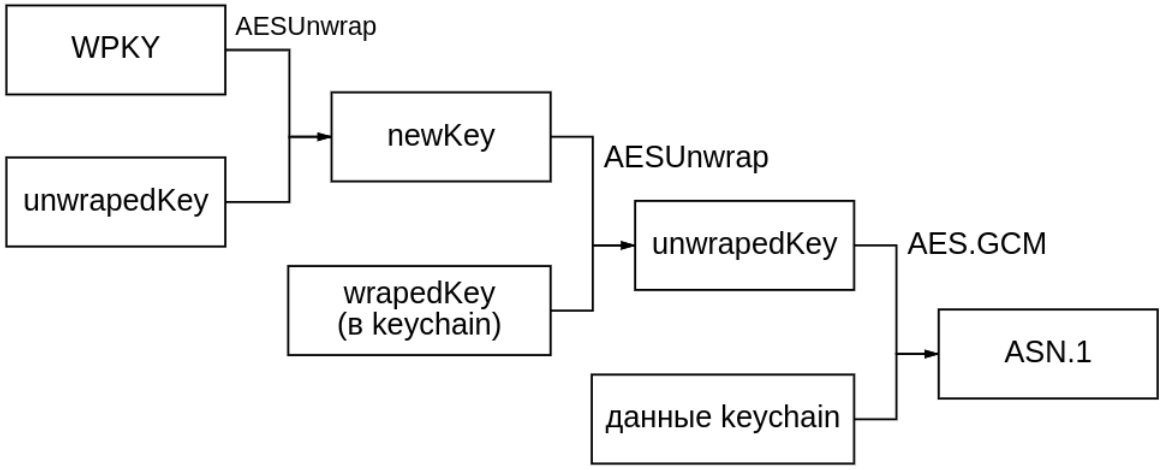

Рис. 3: Дешифровка keychain

5. Реализация

Перед непосредственно отправкой запросов на сервера Apple необходимо пройти процедуру авторизации. Так как модули, с помощью которых она осуществляется, были уже реализованы, то было принято решение переиспользовать их. Сам процесс авторизации подробно описан в работе М. В. Виноградова «Извлечение резервной копии данных устройства на операционной системе iOS из облачного хранилища iCloud» [8]. После ее прохождения клиент получает несколько токенов, которые затем используются в заголовках запросов:

1. XAppleMDHeader
2. XAppleMDMHeader
3. XAppleMDRinfoHeader

Также для запросов на escrow-proxy в заголовках необходимо указывать PET токен (закодированные в Base64 данные, которые содержат различную информацию о пользователе), так как в противном случае сервер в качестве ответа вернет ошибку авторизации. Для того, чтобы определить, для какого именно устройства мы хотим извлечь данные, сначала необходимо декодировать данные из Tombs, так как каждый keybag содержит хэш устройства, которому соответствуют записи keychain. Далее клиент отправляет запросы на escrow-proxy, в результате чего получает BackupBagPassword для выбранного устройства.

За загрузку записей Tomb отвечает класс Sync. Дальнейшая работа с полученными данными осуществляется в классе KeychainExtractor. В методе ProcessTombs производится их парсинг, там же мы получаем список хэшей устройств и для одного выбранного хэша извлекаем BackupBagPassword при помощи класса EscrowedKeys. Этот класс в свою очередь использует EscrowProxyRequestFactory, в котором происходит формирование запросов на escrow, а также классы EscrowOperationRecords и EscrowOperationsRecover для осуществления SRP и извлечения соответствующего BackupBagPassword. На данном

шаге уже получены все необходимые для расшифровки данные, сама расшифровка происходит в методе ParseKeychainData и в классе KeyBag, который отвечает за декодирование поля BackupKeybag, также он генерирует 32 байтовый ключ шифрования. Общая архитектура модуля представлена на рисунке 4.

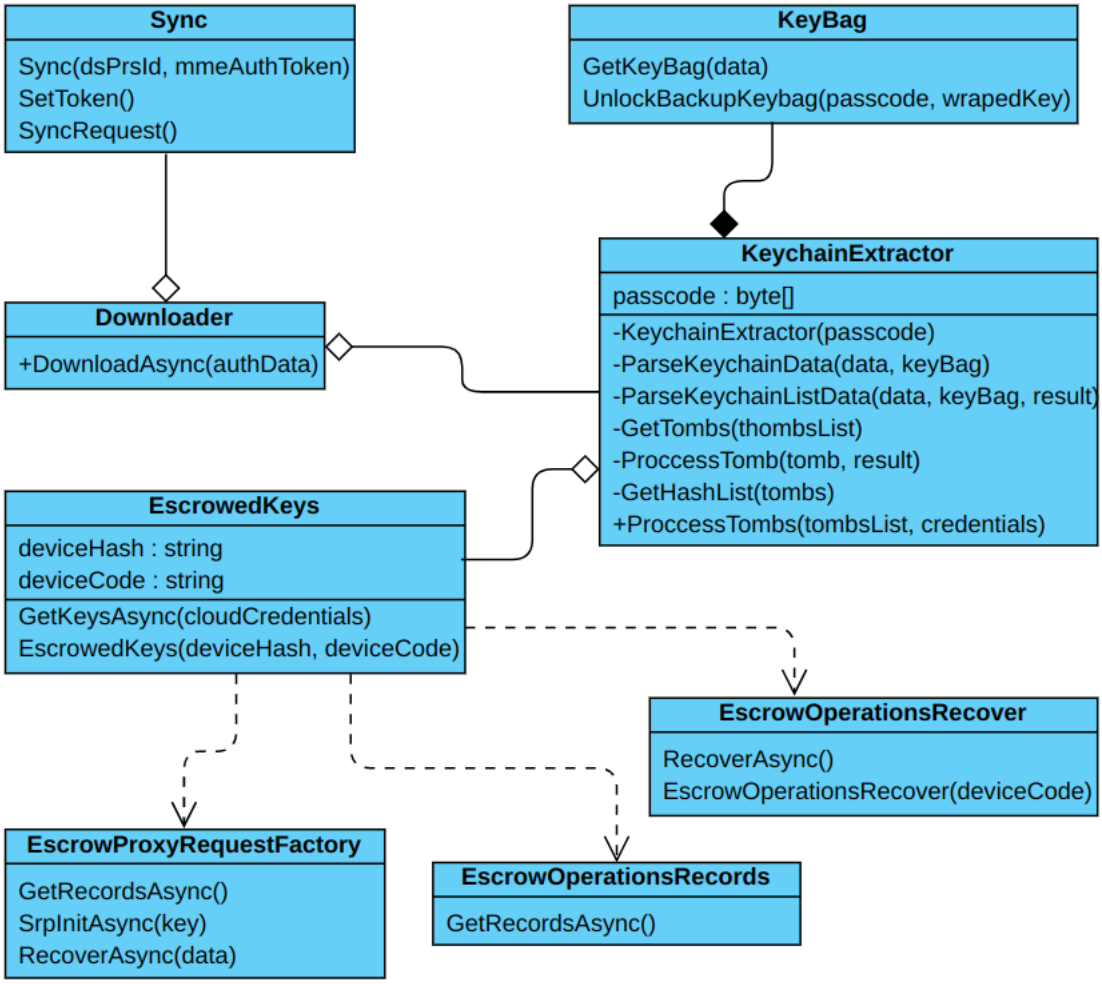

Рис. 4: Общая архитектура модуля

6. Тестирование

Для проверки корректности реализованной программы было проведено ручное тестирование на двух аккаунтах. Были созданы новые учетные записи, в них было авторизовано три тестовых устройства, для каждого из которых было добавлено некоторое количество тестовых данных, которые представляли собой данные авторизации для веб-сайтов, приложений, а также пароли для WiFi-сетей. Само тестирование происходило по следующей схеме: после прохождения двухфакторной авторизации демонстрировался список устройств, авторизованных в учетной записи, далее выбиралось одно из них и вводился код разблокировки этого устройства. После расшифровки данные записывались в специальный файл, и в итоге все данные, загруженные в iCloud, были успешно извлечены. Реализованный модуль только проходит интеграцию в Belkasoft X, поэтому на данный момент данные сохраняются в формате XML. Количество извлеченных записей для разных устройств приведено в Таблицах 1, 2.

	iPhone SE 2020	iPhone 7	iPad Air 4
Веб-сайты	22	10	9
Приложения	10	8	7
WiFi	4	3	3

Таблица 1: Результат извлечения данных для первого аккаунта

	iPhone SE 2020	iPhone 7	iPad Air 4
Веб-сайты	7	10	2
Приложения	12	7	11
WiFi	2	5	1

Таблица 2: Результат извлечения данных для второго аккаунта

7. Заключение

На данный момент были получены следующие результаты:

1. Сделан обзор различных решений.
2. Изучены и применены методы реверсинга для приложений iOS.
3. Изучена схема извлечения зашифрованных данных.
4. Изучена схема дешифровки данных.
5. С помощью языков программирования C, C# реализован прототип, позволяющий извлекать данные Keychain из облачного хранилища iCloud.
6. Проведено ручное тестирование реализованного инструмента.

8. Благодарности

Автор работы благодарит специалиста по вирусной безопасности «Лаборатория Касперского» Михаила Виноградова за всестороннюю консультацию, а также помощь в проведении исследований.

Список литературы

[1] Breaking into the iCloud Keychain, Владимир Каталов – URL: https://hitcon.org/2017/CMT/slide-files/d1_s2_r2.pdf (Date: 20.10.2022)

[2] Devies Andy. Capturing and Decrypting HTTPS Traffic From iOS Apps Using Frida. — URL: https://andydavies.me/blog/2019/12/12/capturing-and-decrypting-https-traffic-from-ios-apps/ (Date: 25.11.2022)

[3] In the depth of iCloud keychain.– URL: https://hackmag.com/uncategorized/in-the-depths-of-icloud-keychain/ (Date: 1.12.2022)

[4] Apple Platform Security. Documentation (Date: 2.12.2022)

[5] Frida Documentation. – URL: https://frida.re/docs/home/ (Date: 10.12.2022)

[6] SSL Killswitch docs – URL: https://github.com/nabla-c0d3/ssl-kill-switch2 (Date: 5.10.2022)

[7] How to Monitor Mobile App Traffic With Sniffers – URL: https://medium.com/p/d7d606013beb#5120 (Date: 30.09.2022)

[8] Выпускная квалификационная работа «Извлечение резервной копии данных устройства на операционной системе iOS из облачного хранилища iCloud», М. В. Виноградов

[9] Apple iCloud inside out iCloud backups, FindMyPhone, document storage, iCloud keychain. Vladimir Katalov, ElcomSoft Co. Ltd.

[10] Frida: function interception – https://frida.re/docs/functions/ (Date: 20.03.2023)

[11] iCloud security overview. — URL: https://support.apple.com/en-us/HT202303 (Date: 12.12.2022)

[12] Corecrypto documentation: Mathematical routines for the NIST prime elliptic curves (March 28, 2008)