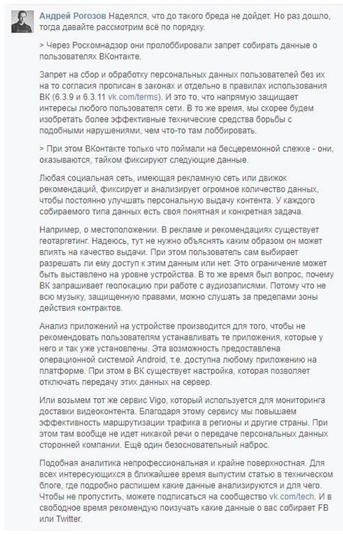
О том, как ВКонтакте собирает информацию о нас: часть 2

Владислав Велюга (vlad805) • July 31, 2017

Перейти к первой части.

Update 2

А еще давайте сразу, вот что ответил (где-то) Андрей Рогозов про данную информацию.



Андрей Рогозов Надеялся, что до такого бреда не дойдет. Но раз дошло, тогда давайте рассмотрим всё по порядку.

> Через Роскомнадзор они пролоббировали запрет собирать данные о пользователях ВКонтакте.

Запрет на сбор и обработку персональных данных пользователей без их на то согласия прописан в законах и отдельно в правилах использования ВК (6.3.9 и 6.3.11 vk.com/terms). И это то, что напрямую защищает интересы любого пользователя сети. В то же время, мы скорее будем изобретать более эффективные технические средства борьбы с подобными нарушениями, чем что-то там лоббировать.

> При этом ВКонтакте только что поймали на бесцеремонной слежке - они, оказываются, тайком фиксируют следующие данные.

Любая социальная сеть, имеющая рекламную сеть или движок рекомендаций, фиксирует и анализирует огромное количество данных, чтобы постоянно улучшать персональную выдачу контента. У каждого собираемого типа данных есть своя понятная и конкретная задача.

Например, о местоположении. В рекламе и рекомендациях существует геотаргетинг. Надеюсь, тут не нужно объяснять каким образом он может влиять на качество выдачи. При этом пользователь сам выбирает разрешать ли ему доступ к этим данным или нет. Это ограничение может быть выставлено на уровне устройства. В то же время был вопрос, почему ВК запрашивает геолокацию при работе с аудиозаписями. Потому что не всю музыку, защищенную правами, можно слушать за пределами зоны действия контрактов.

Анализ приложений на устройстве производится для того, чтобы не рекомендовать пользователям устанавливать те приложения, которые у него и так уже установлены. Эта возможность предоставлена операционной системой Android, т.е. доступна любому приложению на платформе. При этом в ВК существует настройка, которая позволяет отключать передачу этих данных на сервер.

Или возьмем тот же сервис Vigo, который используется для мониторинга доставки видеоконтента. Благодаря этому сервису мы повышаем эффективность маршрутизации трафика в регионы и другие страны. При этом там вообще не идет никакой речи о передаче персональных данных сторонней компании. Ещё один безосновательный наброс.

Подобная аналитика непрофессиональная и крайне поверхностная. Для всех интересующихся в ближайшее время выпустим статью в техническом блоге, где подробно распишем какие данные анализируются и для чего. Чтобы не пропустить, можете подписаться на сообщество vk.com/tech. И в свободное время рекомендую поизучать какие данные о вас собирает FB или Twitter.

>>>

Моя статья о сборе информации о пользователях официального приложения ВКонтакте под Android уж слишком разошлась в Интернете. Я заметил, что не так уж и мало людей относится к конфиденциальности своих данных не наплевательски и приняли материал довольно "тепло".

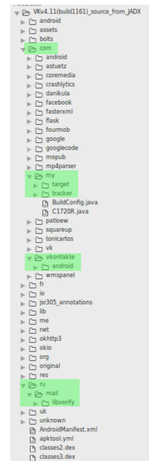
Обсуждения в комментариях под моим постом в ВК тоже оказались довольно интересными. Туда пришел бывший разработчик этого самого приложения **Григорий Клюшников** и разработчик его модификации **Эдуард Безменов**. Оба они рассказали немало интересных вещей. Что ж, мне тоже стало интересно, что еще я не смог наснифить на реальном устройстве и о чем конкретно они говорили.

Наш инструментарий на сегодня

Имеется установочный APK-файл приложения версии 4.12.1. Имеются его исходники (код), полученные путем декомпилирования приложения (грубо говоря, установочный файл был разобран и по нему воссоздан код на программном языке, в нашем случае - Java). Декомпилятор - JADX.

> ***Небольшой ликбез***
> *В Java-проекте (а приложение под Android таковым и является) весь код поделен на классы; классы сгруппированы в папки -- пакеты (пэкедж, package), пакет может находится в другом пакете. Каждое приложение может иметь неограниченное число пакетов, которые могут "общаться", взаимодействовать друг с другом.*

Декомпилировав приложение, мы видим следующее



com.my.tracker - тот самый трекер MailRu, о котором говорил Эдуард и Григорий;
com.my.target - пакет, связанный с трекером; com.vkontakte.android - непосредственно само приложение и ru.mail.libverify - еще один пакет от MailRu

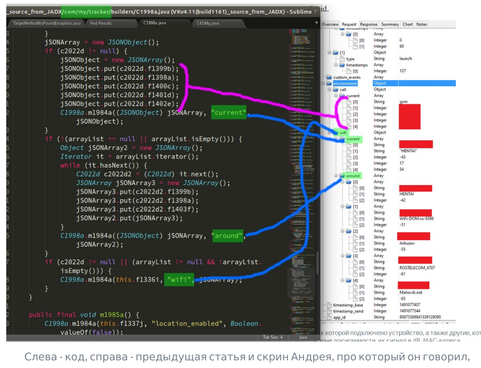
С декомпилированным кодом будет сложнее, поскольку не все переменные и классы будут названы именами, которые будут понятны человеку. В основном это названия чем-то напоминающие шестнадцатеричное число, например, `C1998a.java`. Есть у меня еще одно предположение, что код мог быть специально обфусцирован (*obfuscate* — делать неочевидным, запутанным, сбивать с толку). Но некоторые "зацепки" остаются, по которым можно понять что это за код, за что он отвечает и что именно что делает.

Результаты

Не буду томить, сразу к делу.

com.my.tracker

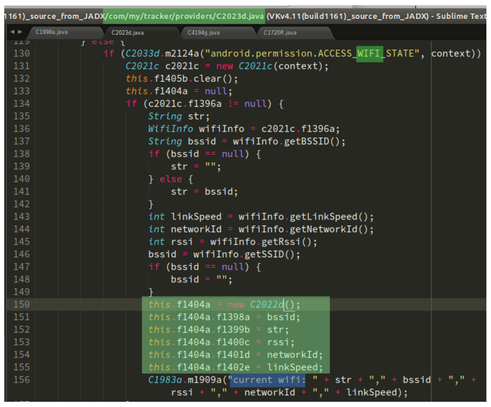
Начнем свое путешествие с файла класса `com.my.tracker.builders.C1998a`..



Слева - код, справа - предыдущая статья и скрин Андрея, про который он говорил,

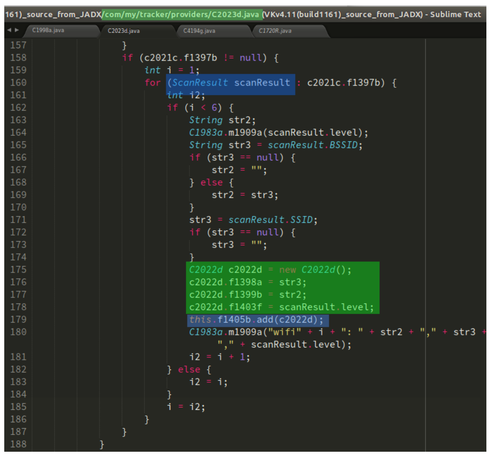
что приложение сливает точки доступа Wi-Fi.

Теперь следим за ходом мысли: в строке 298 (чуть выше строки с "around", сорян, урезал номера в скрине) объявляется переменная c2002d2 с типом **C2022d**, получаемая из итератора (грубо говоря, коллекция/массив/список). Класс **C2022d** определен в пакете **com.my.tracker.providers** в классе **C2023d**. Определение нам его не интересно, ибо оно не содержит понятных человеку названий полей. В этом же классе есть объявление этого класса и заполнение этого класса данными.



Сбор информации о текущей точке доступа.

Вот тут то, по логу после объявления и заполнения экземпляра класса значениями, мы и понимаем, что **C2022d** -- это класс (а если быть правильнее, то структура), хранящий в себе данные о точке доступа Wi-Fi. Чуть ниже собираются данные обо всех окружающих точках.



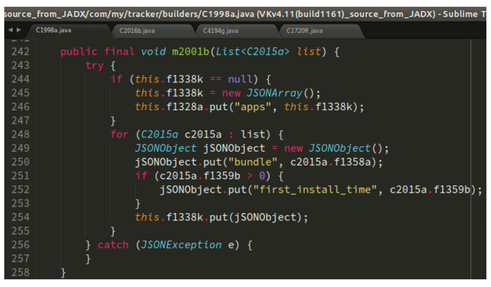
В цикле перебирается результат сканирования (160), создается и заполняется класс C2022d = информация о ТД Wi-Fi (175-178) и добавляется в список (179).

А еще ниже мы видим, как пытаются собрать данные о CID и LAC.

> *<wikipedia>*
> ***Cell ID, CID** — «идентификатор соты». Это параметр, который присваивается оператором каждому сектору каждой базовой станции, и служит для его идентификации.*
> ***LAC, Local Area Code** — код локальной зоны. Локальная зона — это совокупность базовых станций, которые обслуживаются одним BSC — контроллером базовых станций.*
> ***Базовая станция** — совокупность оборудования одного оператора, установленного на одной площадке (вышке, здании) и предназначенного для непосредственной связи сети с мобильными терминалами абонентов*
> *</wikipedia>*

Coarse location - примерное местоположение

Возвращаемся назад к C1998a.java ...



Тоже интересно... apps (приложения), first_install_time (время первой установки)...

... Неужто тоже отправляет список установленных приложений? **C2015a** - еще одна структура в **com.my.tracker.providers.C2016b**. Определение не интересует, использование тут же...

Получение списка приложений (84), создание собственного списка приложений со своими структурами (88), обход всего списка циклом (90), получение информации об одном приложении (91), если оно не системное (пруф), то добавляется в список.

Снова возвращаемся в `C1998a.java`.

```
196 public final void m1996a(String[] strArr) {
197 C1998a.m1984a(this.f1330c, "email", new JSONArray(Arrays.asList(strArr)));
198 }
199
200 public final void m2002b(String[] strArr) {
201 C1998a.m1984a(this.f1330c, "ok_id", new JSONArray(Arrays.asList(strArr)));
202 }
203
204 public final void m2007c(String[] strArr) {
205 C1998a.m1984a(this.f1330c, "vk_id", new JSONArray(Arrays.asList(strArr)));
206 }
207
208 public final void m2010d(String[] strArr) {
209 C1998a.m1984a(this.f1330c, "icq_id", new JSONArray(Arrays.asList(strArr)));
210 }
211
212 public final void m2013e(String[] strArr) {
213 C1998a.m1984a(this.f1330c, "custom_user_id", new JSONArray(Arrays.asList(strArr)));
214 }
```

Здесь же, откуда-то собираются списки электронных адресов, ID одноклассников, ВК, ICQ (*лол*), и другие ID...

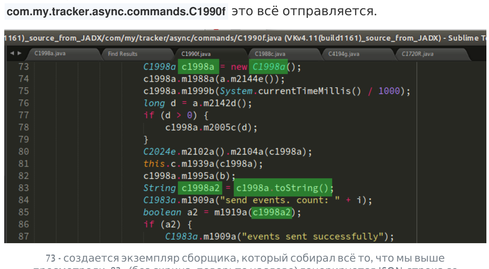
```
public final void m2023m(String str) {
C1998a.m1984a(this.f1332e, "sim_loc", (Object) str);
}

public final void m2024n(String str) {
C1998a.m1984a(this.f1332e, "operator_id", (Object) str);
}

public final void m2025o(String str) {
C1998a.m1984a(this.f1332e, "operator_name", (Object) str);
}

public final void m2026p(String str) {
C1998a.m1984a(this.f1332e, "sim_operator_id", (Object) str);
}

public final void m1994a(String str, String str2) {
C1998a.m1984a(this.f1332e, "connection", (Object) str);
C1998a.m1984a(this.f1332e, "connection_type", (Object) str2);
}

public final void m2008d(int i) {
C1998a.m1984a(this.f1332e, "bluetooth_enabled", Integer.valueOf(i));
}
```

... sim_loc (?), ID и имя оператора, информация о соединении и его типе (Wi-Fi/мобильная связь), включенности BlueTooth...

```
public final void m2012e(String str) {
C1998a.m1984a(this.f1331d, "device", (Object) str);
}

public final void m2015f(String str) {
C1998a.m1984a(this.f1331d, "os", (Object) str);
}

public final void m2017g(String str) {
C1998a.m1984a(this.f1331d, "manufacture", (Object) str);
}

public final void m2018h(String str) {
C1998a.m1984a(this.f1331d, "mac", (Object) str);
}

public final void m2019i(String str) {
C1998a.m1984a(this.f1331d, "osver", (Object) str);
}

public final void m2020j(String str) {
C1998a.m1984a(this.f1331d, "lang", (Object) str);
}

public final void m2021k(String str) {
C1998a.m1984a(this.f1331d, "timezone", (Object) str);
}
```

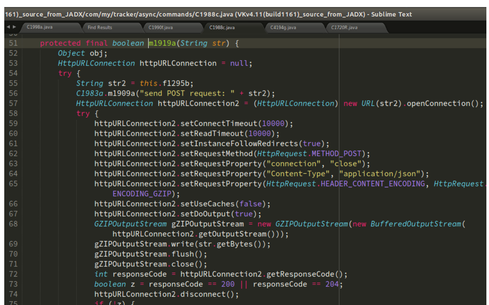
..., информация об устройстве, названии и версии ОС на устройстве, производителе, MAC-адресе устройства, языке, часовом поясе, ... и еще пара мелочей.

И вот собрали мы это всё... А зачем? В классе `com.my.tracker.async.commands.C1990f` это всё отправляется.



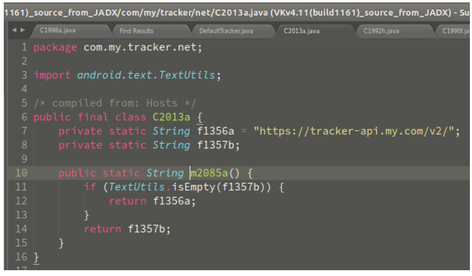
73 - создается экземпляр сборщика, который собирал всё то, что мы выше просмотрели. 83 - (без скрина, поверьте наслово) генерируется JSON-строка со всеми данными, 85 - данные отправляются...

Метод `m1919a` ... где же он? Он в родительном классе `C1988c`



Грубо говоря - здесь делается запрос в интернет по адресу... указанному в конструкторе класса C1988c, но он же не создавался выше!

Глубоко порывшись в поиске по всему проекту (по названию поля `f1295b`, по унаследованным классам), пришел к методу `C2013a.m2085a`..

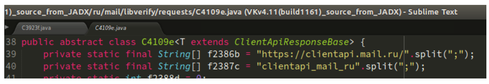


А вот и адрес, куда отправлялась информация

Всё, на этом хватит с MyTracker...

ru.mail.libverify

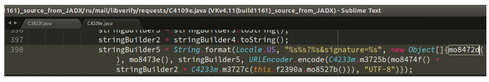
ru.mail.ibverify.requests.C4109e



Класс C4109e, в самом начале строка, содержащая домен mail.ru, куда будут делаться запросы.

Прямо в этом же классе (см. заголовок Sublime) C4109e содержатся данные о IMEI, IMSI, телефоне, операторе, широте/долготе и точности местоположения.

В C4109e: метод mo8472d выбирает хост из f2386b (через один скрин выше), куда отправлять данные



В C4109e: в методе m3158u формируется полный URL-адрес для запроса, используя метод mo8472d, который описан выше.

На этом пока всё.

Итог

Приложение ВКонтакте для Android помимо своих метрик и телеметрий отправляет такой же, и даже больший, объем данных для третьих лиц: MyTracker и MailRu LibVerify.

Обсуждение

В комментах под постом у меня на стене.