**Декомпиляция и отладка Android-приложений**

В статье будет рассказано о том, как подключить отладчик к Android-приложению и пошагово пройти через вызываемые методы, используя информацию, полученную после декомпиляции приложения.

Автор: Eric Gruber

В статье будет рассказано о том, как подключить отладчик к Android-приложению и пошагово пройти через вызываемые методы, используя информацию, полученную после декомпиляции приложения. Хорошая новость в том, что для отладки не требуются привилегии суперпользователя. Описанные техники могут быть очень кстати во время пентестов мобильных приложений, поскольку мы можем «проникнуть» в код во время работы программы, получить и записать информацию, к которой обычно у нас нет доступа. Например, можно перехватить трафик перед шифрованием и на лету получить ключи, пароли и любую другую ценную информацию. Статья будет полезна пентестерам и разработчикам мобильных приложений, желающих получить более глубокие знания о возможных атаках на платформе Android.

Требования к тестовой среде:

* Операционная система: Windows / Mac OS X / Linux
* Java (рекомендуется версия 1.7)
* IDE (Eclipse, IntelliJ IDEA, Android Studio)
* Android SDK ([https://developer.android.com/sdk/index.html?hl=i](http://www.securitylab.ru/bitrix/exturl.php?goto=https%3A%2F%2Fdeveloper.android.com%2Fsdk%2Findex.html%3Fhl%3Di))
* APKTool ([https://code.google.com/p/android-apktool/](http://www.securitylab.ru/bitrix/exturl.php?goto=https%3A%2F%2Fcode.google.com%2Fp%2Fandroid-apktool%2F))/APK Studio ([http://apkstudio.codeplex.com](http://apkstudio.codeplex.com/))
* Устройство / эмулятор на базе Android
* Dex2Jar ([https://code.google.com/p/dex2jar/](http://www.securitylab.ru/bitrix/exturl.php?goto=https%3A%2F%2Fcode.google.com%2Fp%2Fdex2jar%2F))
* JD-GUI ([http://jd.benow.ca/](http://www.securitylab.ru/bitrix/exturl.php?goto=http%3A%2F%2Fjd.benow.ca%2F))

В статье будет использоваться следующая конфигурация: Windows 8, Android Studio и IntelliJ IDEA. Устройство: Nexus 4 с Android версии 4.4.4. Рекомендую все утилиты добавить в переменную окружения PATH, чтобы облегчить и ускорить доступ к этим инструментам.

Android application package (APK), используемый в статье, можно скачать отсюда:[com.netspi.egruber.test.apk](http://www.securitylab.ru/bitrix/exturl.php?goto=https%3A%2F%2Fblog.netspi.com%2Fwp-content%2Fuploads%2F2015%2F01%2Fcom.netspi.egruber.test_.apk).

Настройка устройства

Инструкция ниже поможет вам подготовить устройство для экспериментов.

Активация раздела Developer Options

Для начала на Android-устройстве должна быть разрешена отладка через USB (опция USB debugging), что позволит «общаться» с девайсом при помощи инструментов из набора Android SDK. Однако перед этим необходимо активировать раздел Developer options. На устройстве зайдите в раздел Settings > About Phone и кликните несколько раз на пункт Build Number, после чего должно появиться сообщение о том, что раздел Developer options активирован.

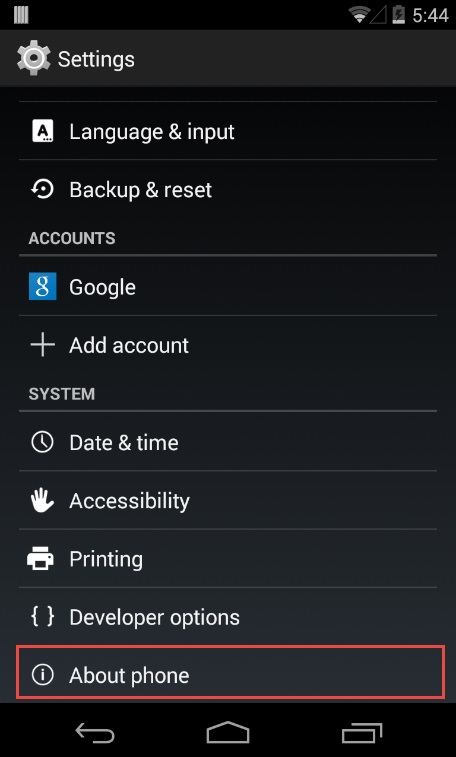




Рисунок 1: Для того чтобы активировать раздел Developer options, необходимо несколько раз кликнуть на Build number

Разрешение отладки через USB

Чтобы разрешить отладку через USB-порт, зайдите в раздел Settings > Developer options и отметьте флажок напротив USB debugging.

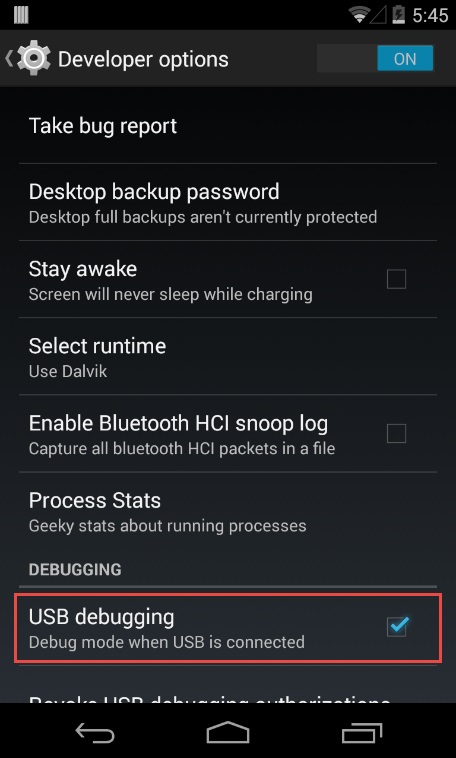


Рисунок 2: Включение опции USB debugging

Подключение устройства и запуск ADB

После подключение устройства к компьютеру через USB-порт, должно появиться сообщение «USB debugging connected on the device». Также следует проверить, можно ли подключиться к устройству при помощи приложения Android Debug Bridge (ADB), входящего в состав Android SDK (пакет Android SDK Platform-tools). В командной строке введите следующую команду:

adb devices

Устройство должно отобразиться в списке.

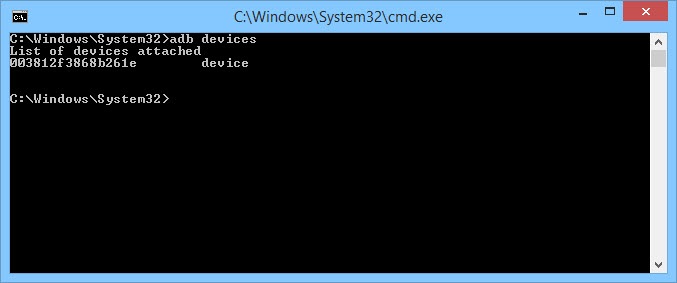


Рисунок 3: Список подключенных устройств

Если устройство не отобразилось в списке, то наиболее вероятная причина в некорректно установленных драйверах (в Windows). В зависимости от устройства драйвер можно найти либо в Android SDK, либо на сайте производителя.

Проверка приложения на возможность отладки

Перед отладкой Android-приложений вначале необходимо проверить, есть ли такая возможность. Проверку можно выполнить несколькими способами.

Первый способ – запустить Android Device Monitor, входящий в состав Android SDK (в папке tools). В Windows файл называется monitor.bat. При открытии Android Device Monitor устройство отобразится в разделе Devices.

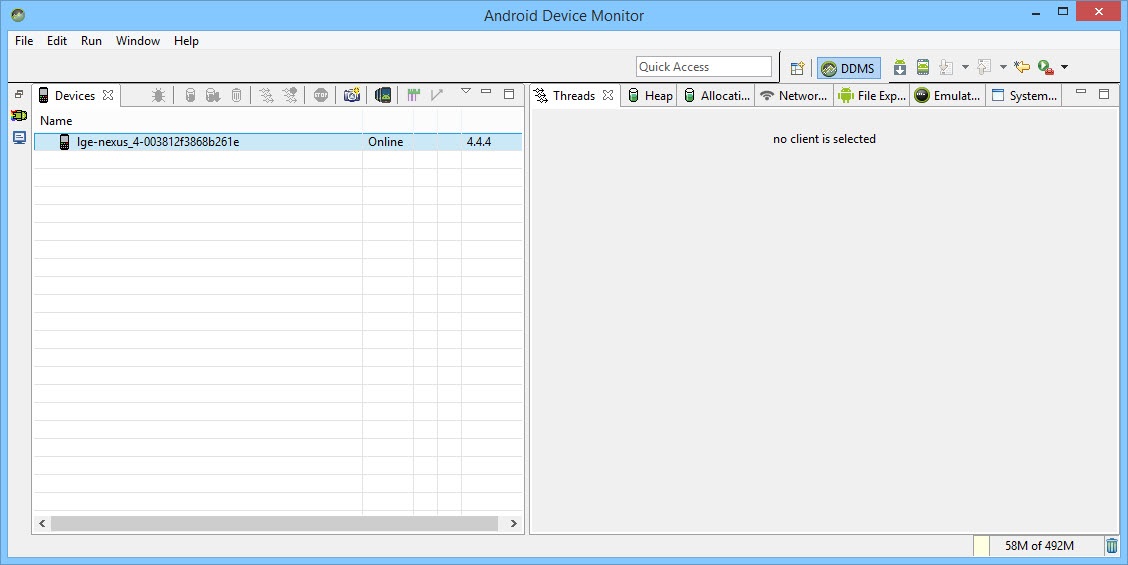
[](http://www.securitylab.ru/_Article_Images/2015/analytics/01/eric-4.jpg)

Рисунок 4: Приложение Android Device Monitor

Если какое-либо приложение на устройстве можно отлаживать, это приложение также отобразится в списке. Я создал тестовую программу, но список пуст, поскольку программу отлаживать нельзя.

Второй способ проверить приложение на возможность отладки – исследовать файл AndroidManifest.xml из пакета приложения (APK, Android application package). APK представляет собой zip-архив, содержащий всю информацию, необходимую для запуска приложения на Android-устройстве.

Всякий раз, когда приложения загружается из Google Play Store, также загружается и пакет приложения. Все загруженные APK-файлы обычно хранятся на устройстве в папке /data/app. Если у вас нет прав суперпользователя, вы не сможете получить список файлов из директории /data/app. Хотя, если вы знаете имя APK-файла, можете скопировать его при помощи утилиты adb. Чтобы узнать имя APK-файла, введите следующую команду:

adb shell

Появится командная строка устройства. Затем введите следующую команду:

pm list packages -f

Отобразится список всех пакетов на устройстве.

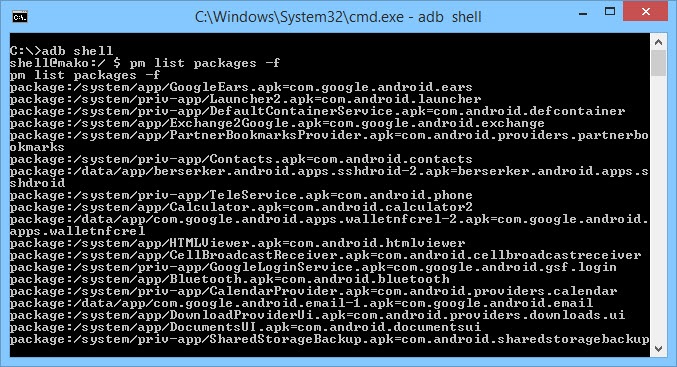


Рисунок 5: Перечень пакетов на устройстве

Глядя на список, находим тестовое приложение.

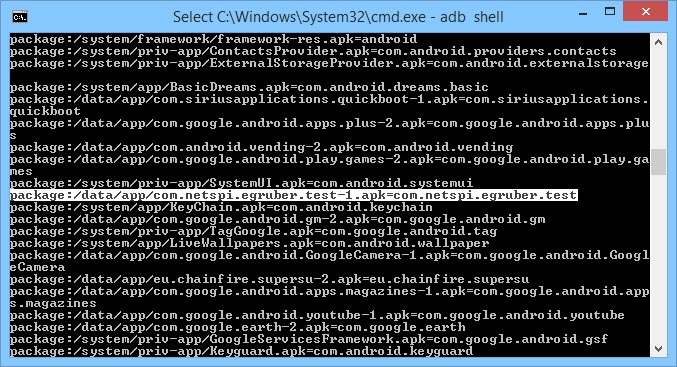


Рисунок 6: Пакет созданного тестового приложения (выделено белым)

Теперь необходимо скопировать файл пакета. Открываем шелл и вводим следующую команду:

adb pull /data/app/[.apk file] [location]

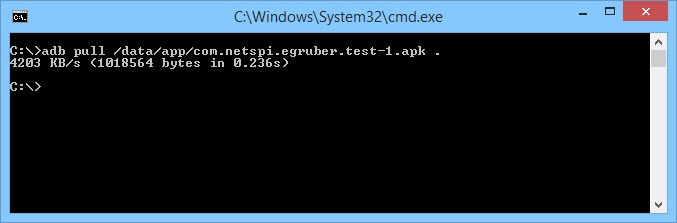


Рисунок 7: Копируем APK-файл с устройства в систему

Теперь нужно открыть файл пакета и исследовать содержимое AndroidManifest.xml. К сожалению, мы не можем просто так распаковать архив, поскольку APK-файл закодирован в бинарном формате. Для раскодировки чаще всего используется утилита apktool, хотя я использую APK Studio, поскольку у этого приложения дружелюбный графический интерфейс. Далее в статье будет рассказываться об APK Studio.

В APK Studio кликните на маленькую зеленую иконку, задайте имя проекту и укажите путь к APK файлу. Затем укажите пусть для сохранения проекта.

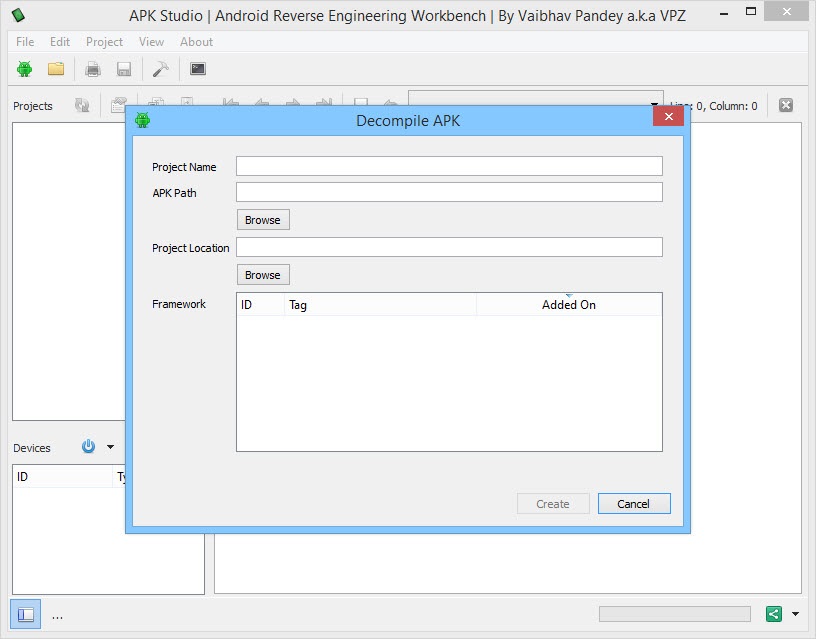


Рисунок 8: Создание нового проекта в APK Studio

После открытия APK выберите файл AndroidManifest.xml и посмотрите параметры тега application. Если флаг android:debuggable отсутствует (или присутствует, но установлено значение false), значит, приложение отлаживать нельзя.

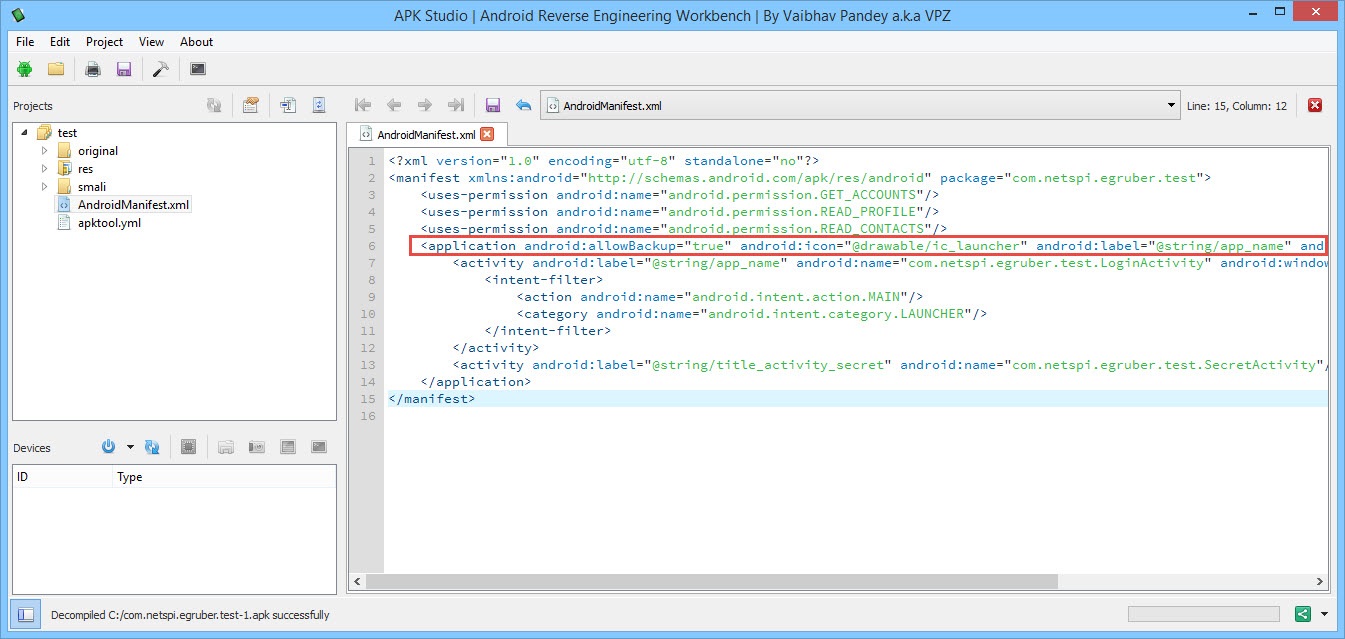
[](http://www.securitylab.ru/_Article_Images/2015/analytics/01/eric-9.jpg)

Рисунок 9: Содержимое файла AndroidManifest.xml

Модификация файла AndroidManifest.xml

При помощи утилиты apktool или APK Studio мы можем модифицировать файлы и упаковывать содержимое обратно в пакет. Сейчас мы изменим файл AndroidManifest.xml так, чтобы приложение можно было отлаживать. Добавляем внутрь тега application строчку android:debuggable="true".

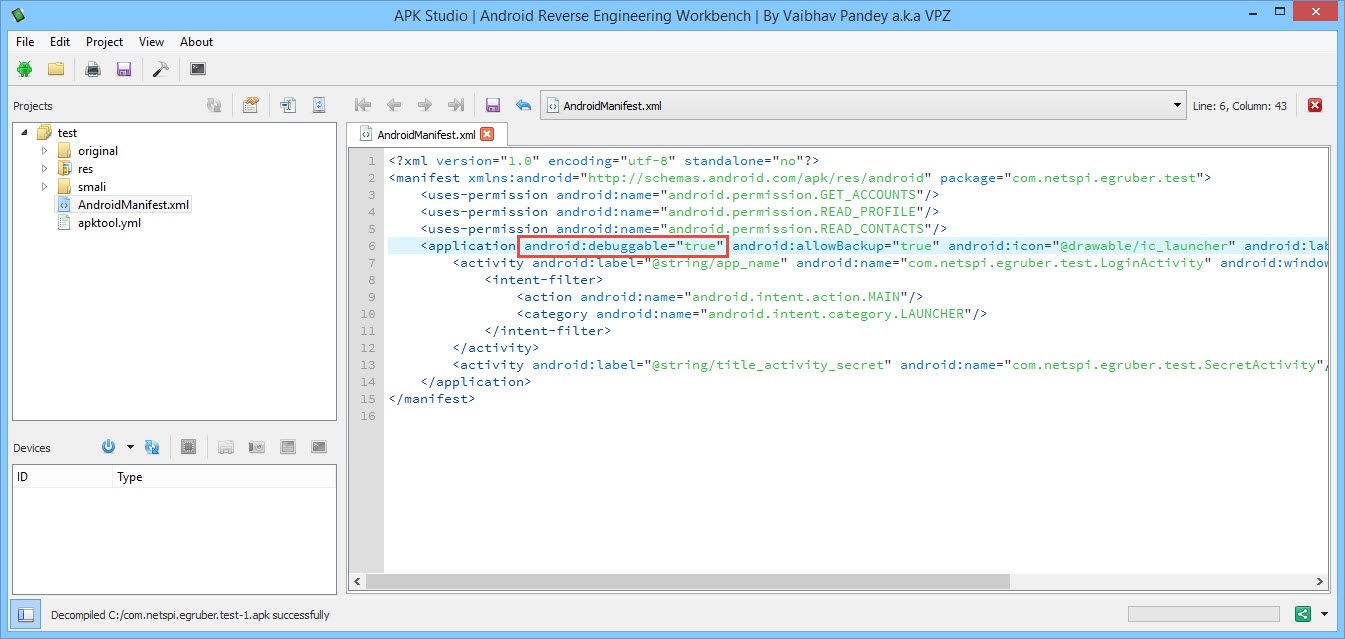
[](http://www.securitylab.ru/_Article_Images/2015/analytics/01/eric-10.jpg)

Рисунок 10: Изменяем содержимое тега application

После добавления флага кликаем на иконку «молоток» и заново собираем пакет. Пересобранный пакет будет находиться в директории build/apk.

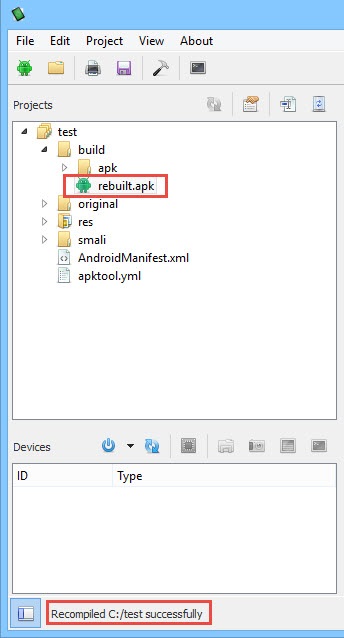


Рисунок 11: Повторная сборка пакета завершилась успешно

После пересборки пакет подписывается и его можно заново установить на устройстве (все Android-приложения должны быть подписаны). Большинство приложений не проверяет сертификат, при помощи которого происходит подпись. В противном случае необходимо изменить код, выполняющий эту проверку.

Теперь нужно установить пересобранный пакет. Вначале удаляем старое приложение при помощи следующей команды:

adb pm uninstall[package name]

Затем устанавливаем новый пакет:

adb install [.apk file]

Также можно удалить и установить пакет одной командой:

adb install -r [.apk file]

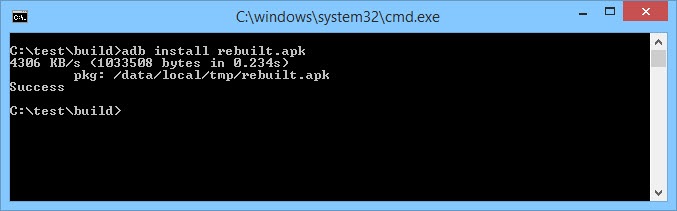


Рисунок 12: Установка пересобранного пакета

Проверьте, чтобы переустановленное приложение корректно запускалось на устройстве. Если все работает, переходим обратно в Android Device Monitor, где должно появиться тестовое приложение.

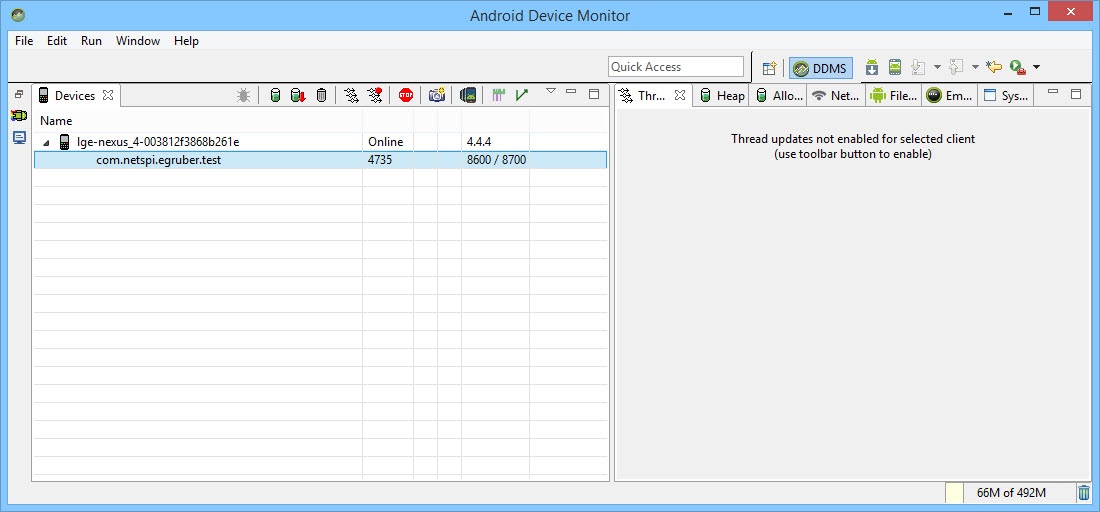
[](http://www.securitylab.ru/_Article_Images/2015/analytics/01/eric-13.jpg)

Рисунок 13: Теперь пересобранное приложение можно отлаживать

Настройка среды разработки (IDE)

Теперь к пересобранному приложению можно подцепить отладчик, но вначале нужно создать проект в среде разработки (в статье используется IntelliJ IDEA). Создаем новый проект. В поле Application name указываем произвольное имя. В поле Package name указываем имя, в точности совпадающее с иерархией папок пересобранного пакета.

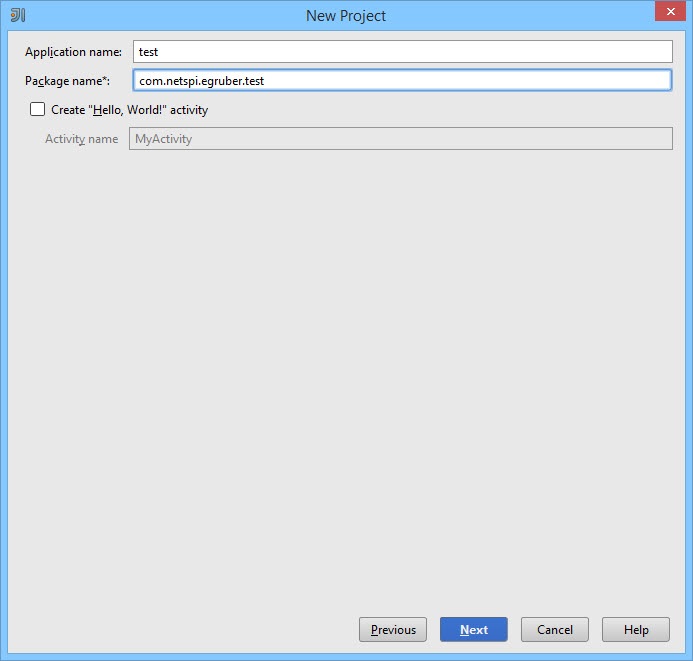


Рисунок 14: Создание нового проекта в IntelliJ IDEA

Обычно имя APK-файла совпадает со структурой папок, хотя, если вы не уверены, в APK Studio проверьте иерархию директорий до папки, где находятся файлы приложений. В моем случае имя и структура папок полностью совпадают (com.netspi.egruber.test).

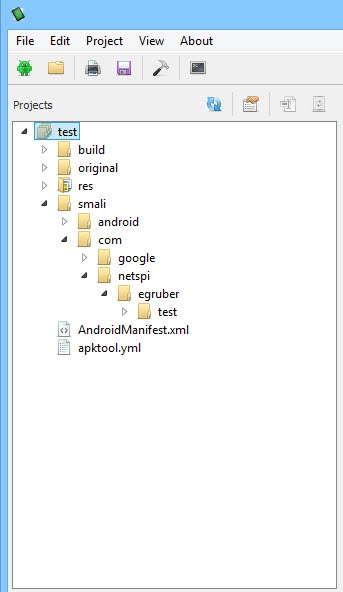


Рисунок 15: Иерархия директорий тестового приложения

Снимите флажок «Create Hello World Activity» и завершите создание проекта (все остальные параметры остаются по умолчанию). Новый проект должен выглядеть примерно так:

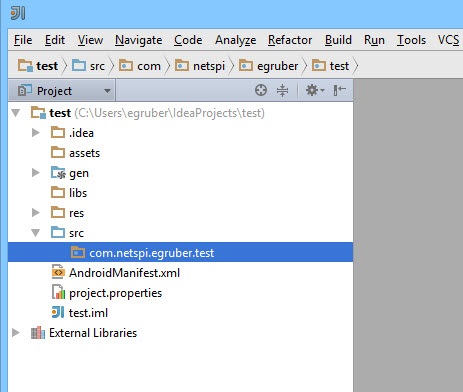


Рисунок 16: Иерархия папок и файлов нового проекта

После создания проекта нужно добавить исходный код из APK-файла для того, чтобы отладчик «знал» имена символов, методов, переменных и т. д. Хорошая новость в том, что Android-приложения можно декомпилировать практически без потери качества (исходный код будет совпадать с оригиналом). После декомпиляции исходный текст импортируется в среду разработки (IDE).

Получение исходных текстов из пакета приложения

Для начала необходимо преобразовать APK в jar-файл. Затем мы при помощи java-декомпилятора получим исходный текст приложения. Преобразование в jar будем делать при помощи утилиты dex2jar. У dex2jar есть файл d2j-dex2jar.bat, используемый для конвертирования APK в jar. Синтаксис команды довольно прост:

d2j-dex2jar.bat [.apk file]

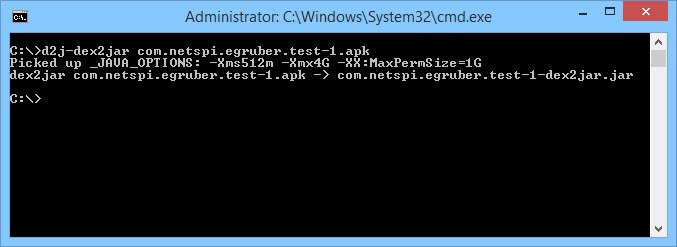


Рисунок 17: Преобразование APK в jar

Затем открываем или перетаскиваем полученный файл в JD-GUI (это java-декомпилятор).

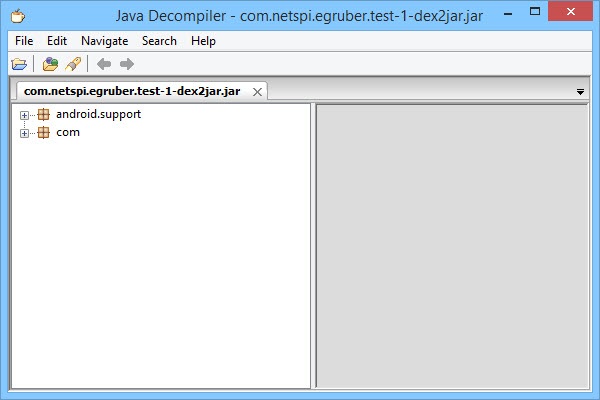


Рисунок 18: Структура jar-файла

Jar-файл должен отобразиться в виде иерархической структуры, внутри которой находятся java-файлы с читабельным исходным кодом. Заходим в File > Save All Sources, чтобы упаковать все исходные тексты в zip-архив.

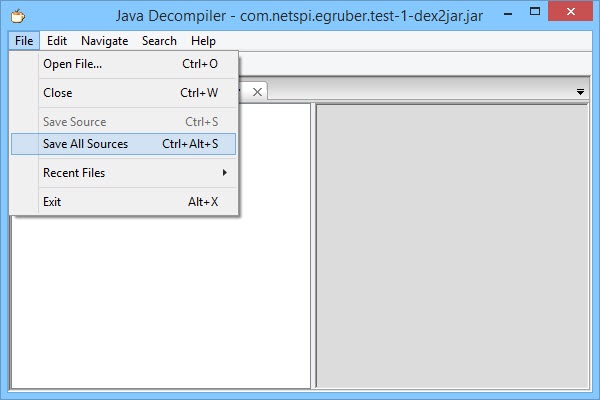


Рисунок 19: Сохранение исходных текстов декомпилированного файла

После сохранения исходных текстов распаковываем архив в отдельную директорию.

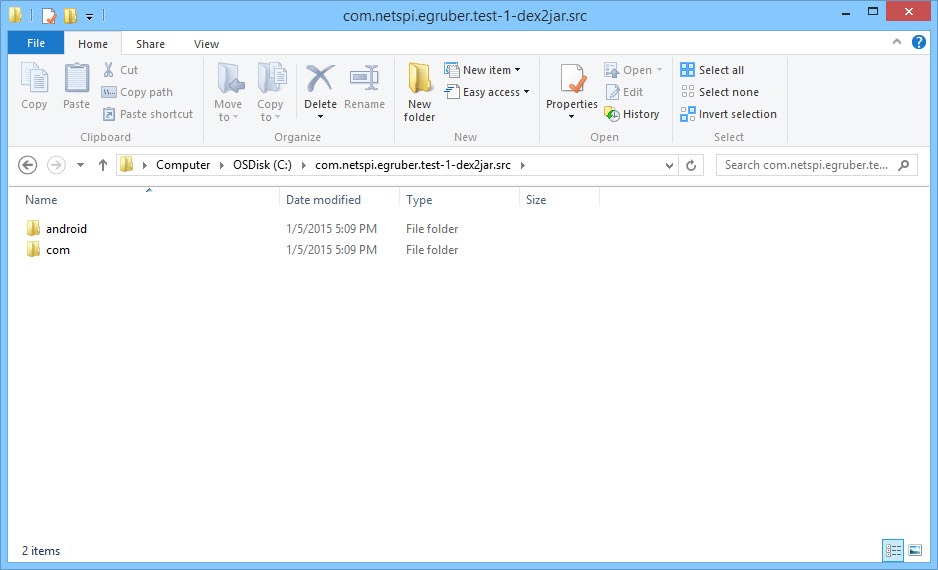
[](http://www.securitylab.ru/_Article_Images/2015/analytics/01/eric-20.jpg)

Рисунок 20: Распакованный архив

Теперь нужно импортировать обе директории в созданный ранее проект в IDE. В IntelliJ заходим в папку src и копируем туда содержимое распакованного архива (две директории).

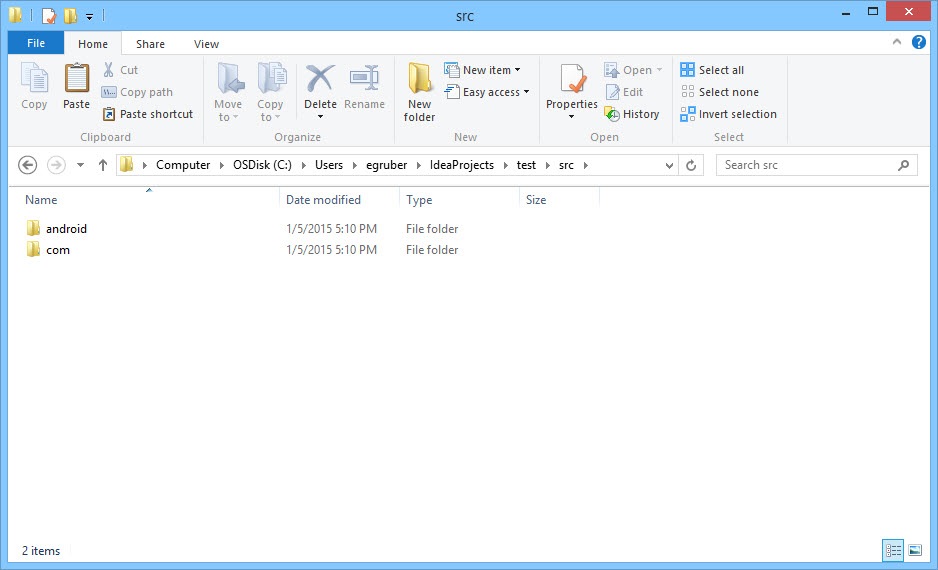
[](http://www.securitylab.ru/_Article_Images/2015/analytics/01/eric-21.jpg)

Рисунок 21: Обе папки скопированы в директорию src

Возвращаясь в Intellij, видим обновленный проект.

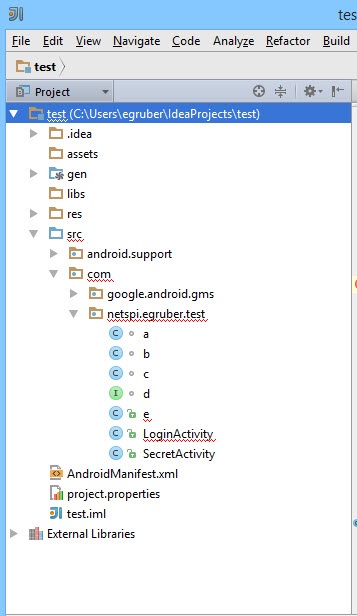


Рисунок 22: В проекте появились исходные тексты

Если мы кликнем на какой-нибудь элемент из списка, то увидим исходный текст. Как видно на скриншоте ниже (исходный текст класса LoginActivity), исходный код обфусцирован при помощи ProGuard.

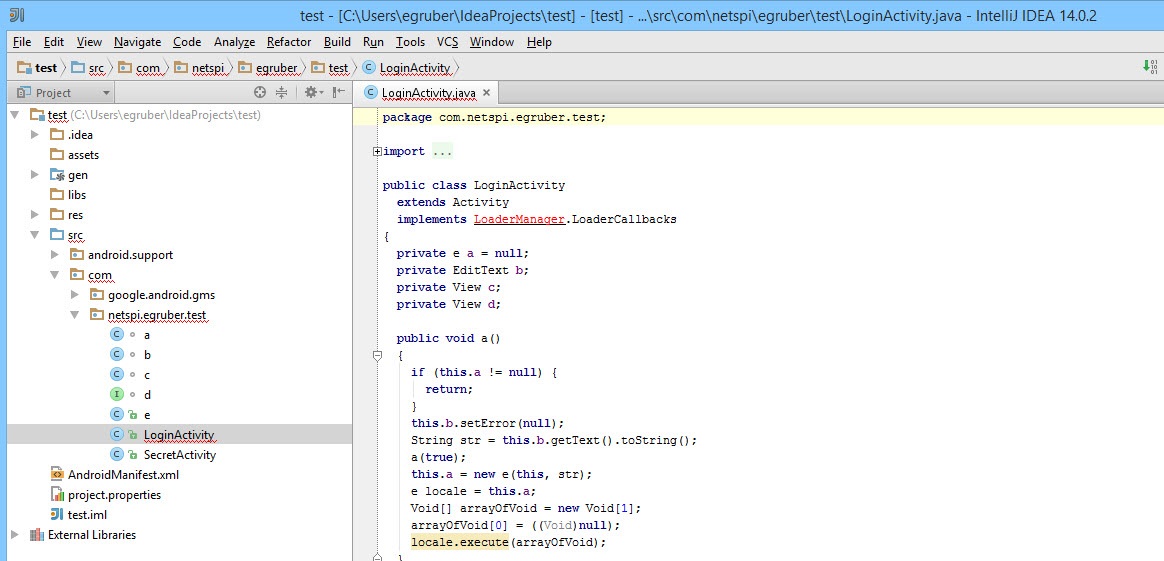
[](http://www.securitylab.ru/_Article_Images/2015/analytics/01/eric-23.jpg)

Рисунок 23: Обфусцированный исходный текст класса LoginActivity

Подключение отладчика

Теперь, когда в проекте появились исходные тексты, мы можем начать устанавливать точки останова на методах и переменных. По достижению точек останова приложение будет останавливаться. В качестве примера я установил точку останова на методе (прямо в обфусцированном коде), отвечающим за обработку информации, введенной в текстовом поле.

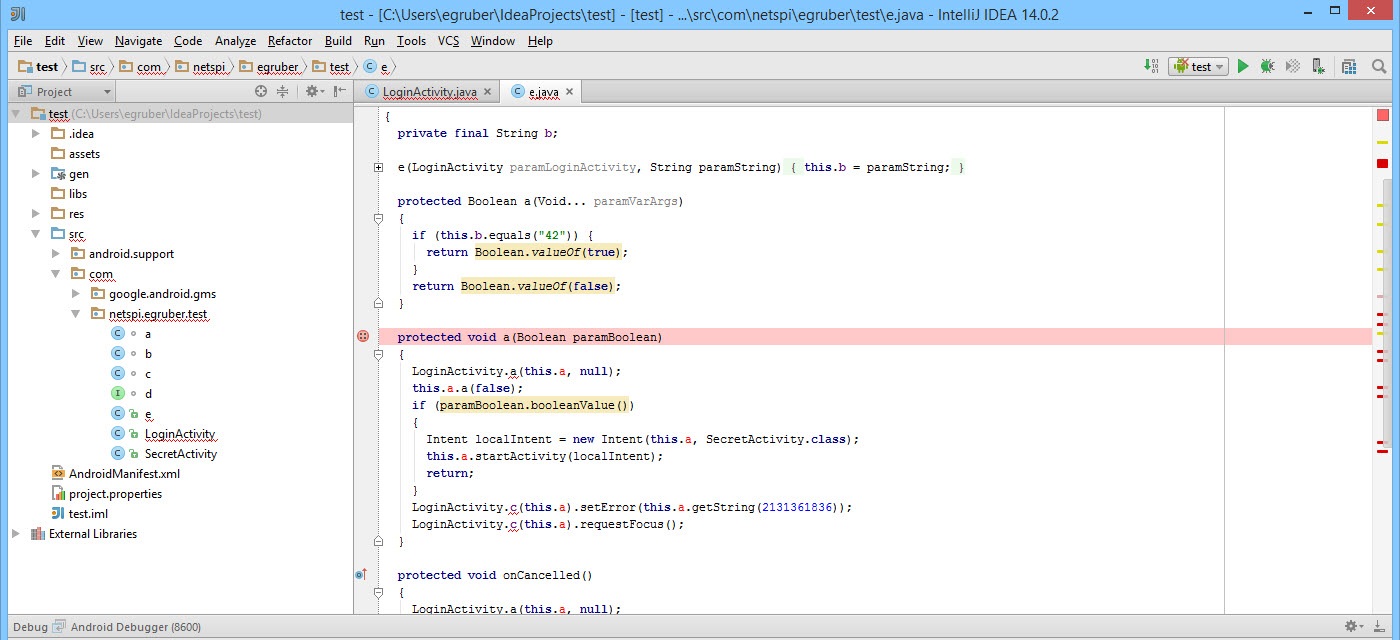
[](http://www.securitylab.ru/_Article_Images/2015/analytics/01/eric-24.jpg)

Рисунок 24: Поставлена точка останова на обфусцированный метод

Как только появилась точка останова, подключаем отладчик к процессу на устройстве, кликнув на иконку с экраном в правом верхнем углу (на вашей IDE иконка может отличаться).

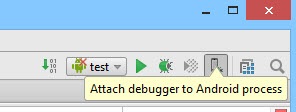


Рисунок 25: Подключаем отладчик к процессу

Далее вам будет предложено выбрать процесс, к которому нужно подключиться. Будут отображены только процессы с флагом android:debuggable="true".

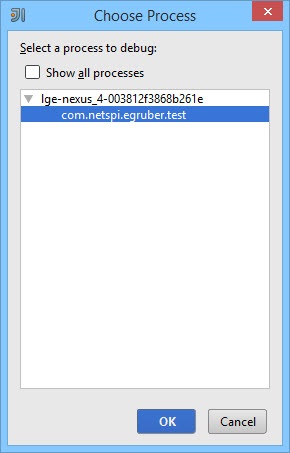


Рисунок 26: Перечень процессов для подключения отладчика

После выбора процесса отладчик подсоединится к устройству.

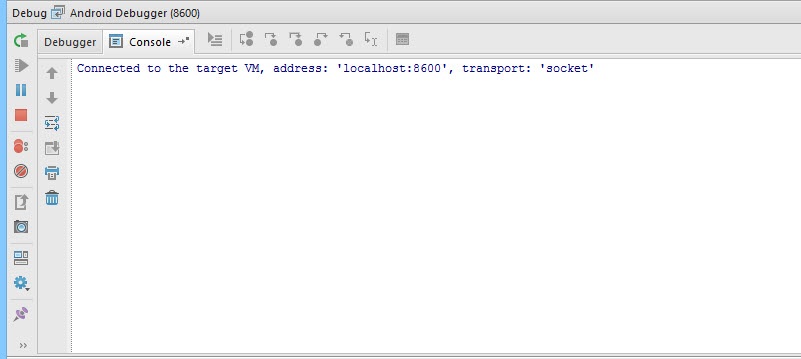


Рисунок 27: Отладчик подключен к процессу, запущенному на устройстве

В текстовое поле я буду вводить число 42 (если помните, на соответствующем методе стоит точка останова).

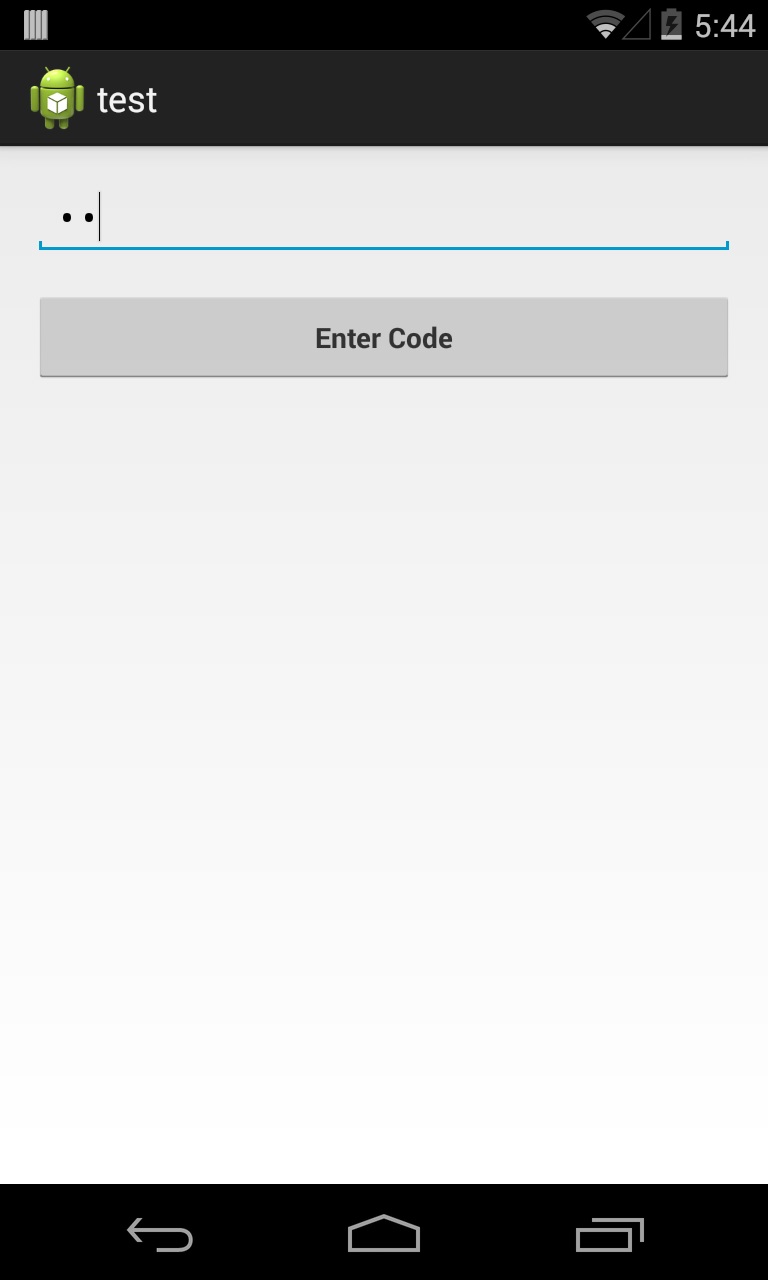
[](http://www.securitylab.ru/bitrix/exturl.php?goto=https://blog.netspi.com/wp-content/uploads/2015/01/img_54ab24710eb8e.png)

Рисунок 28: В текстовое поле вводим число 42

После нажатия на кнопку «Enter Code» выполнение приложения прервется на точке останова, поскольку отладчик «осведомлен», какой метод вызывается на устройстве. Скомпилированное Android-приложение содержит отладочную информацию (например, имена переменных), доступную любому отладчику, совместимому с Java Debug Wire Protocol (JDWP). Если в приложении разрешена отладка, отладчик, совместимый с JDWP (в эту категорию попадает большинство отладчиков идущих в составе сред разработки для Java), сможет подсоединиться к виртуальной машине Android-приложения, а затем считывать и выполнять отладочные команды.

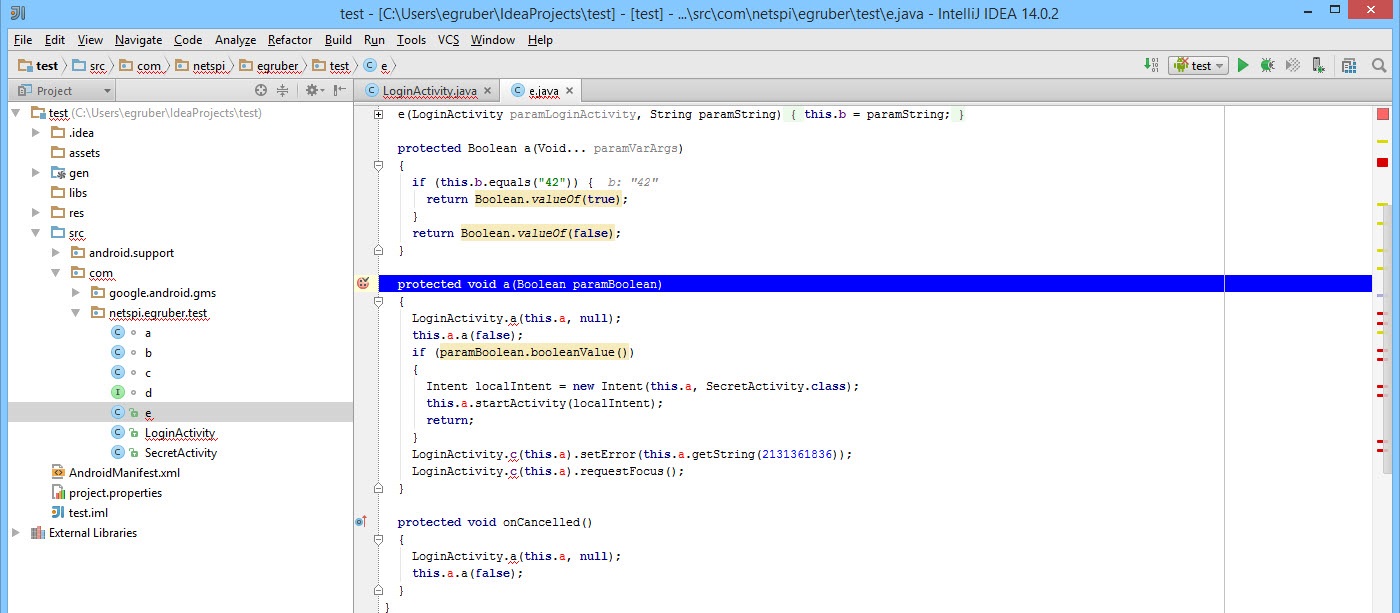
[](http://www.securitylab.ru/_Article_Images/2015/analytics/01/eric-29.jpg)

Рисунок 29: Сработала точка останова

На скриншоте ниже видно число, которое ранее мы ввели в текстовом поле.

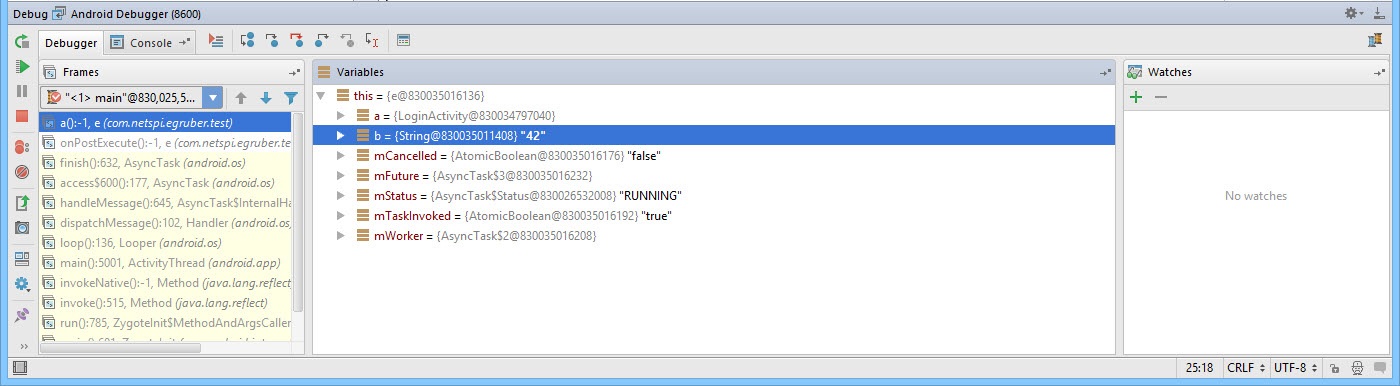
[](http://www.securitylab.ru/_Article_Images/2015/analytics/01/eric-30.jpg)

Рисунок 30: Перечень переменных текущего экземпляра класса

Заключение

Мы можем не только считывать данные в приложении, но и вставлять свои собственные. Это может быть полезно, если мы, например, захотим прервать поток выполнения кода и обойти некоторые участки алгоритма. При помощи отладчика мы можем лучше понять логику работы приложения и проанализировать то, что недоступно обычному пользователю. Например, может оказаться очень полезным просмотр используемых функций шифрования и динамические ключи. Кроме того, иногда при отладке полезно знать, как функции взаимодействуют с файловой системой или базой данных, чтобы понять какая информация сохраняется приложением. Подобные манипуляции доступны на любом Android-устройстве без привилегий суперпользователя.