237.17
Рейтинг

Digital Security

Безопасность как искусство

kerbyj 13 сентября в 07:01

Что мы используем для анализа Android-приложений

Блог компании Digital Security, Информационная безопасность*, Разработка под Android*, Тестирование мобильных приложений*

Всем привет! В этой статье расскажем про инструментарий для анализа мобильных приложений, который мы используем каждый день. Для начала поговорим про то, как запускать мобильные приложения, чем смотреть трафик, а также рассмотрим инструменты для статического и динамического анализа мобильных приложений.

Девайсы

Настоящее устройство

В Digital Security примерно 99% аудитов проводятся на настоящих девайсах. Это избавляет от множества проблем. С ходу могу назвать следующие проблемы, которые решает использование реального устройства:

- Проще пройти аттестацию среды, если таковая реализована в приложении
- Не тратятся полезные ресурсы на рабочей машине
- Встречаются приложения, у которых некоторые функциональности вынесены в нативные библиотеки, и при этом они скомпилированы только для архитектуры ARM. Большинство виртуальных девайсов — это x86_64 или x64; в Android Studio есть ARM-девайс, но версия Android и быстродействие оставляют желать лучшего
- Быстродействие эмулятора всегда ниже, чем у реального устройства

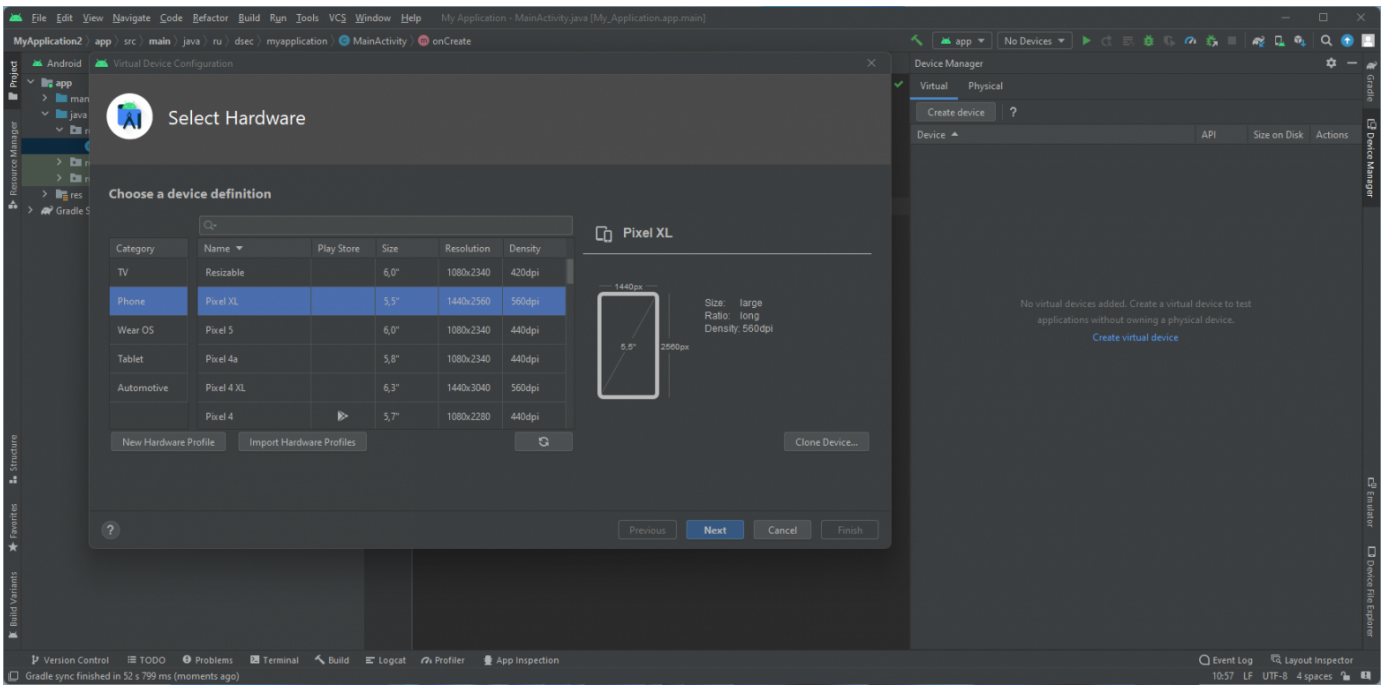
В основном мы используем Samsung с Android 12, пару Xiaomi и пикселей. Также недавно у нас появился планшет на Harmony OS.

Виртуальный девайс

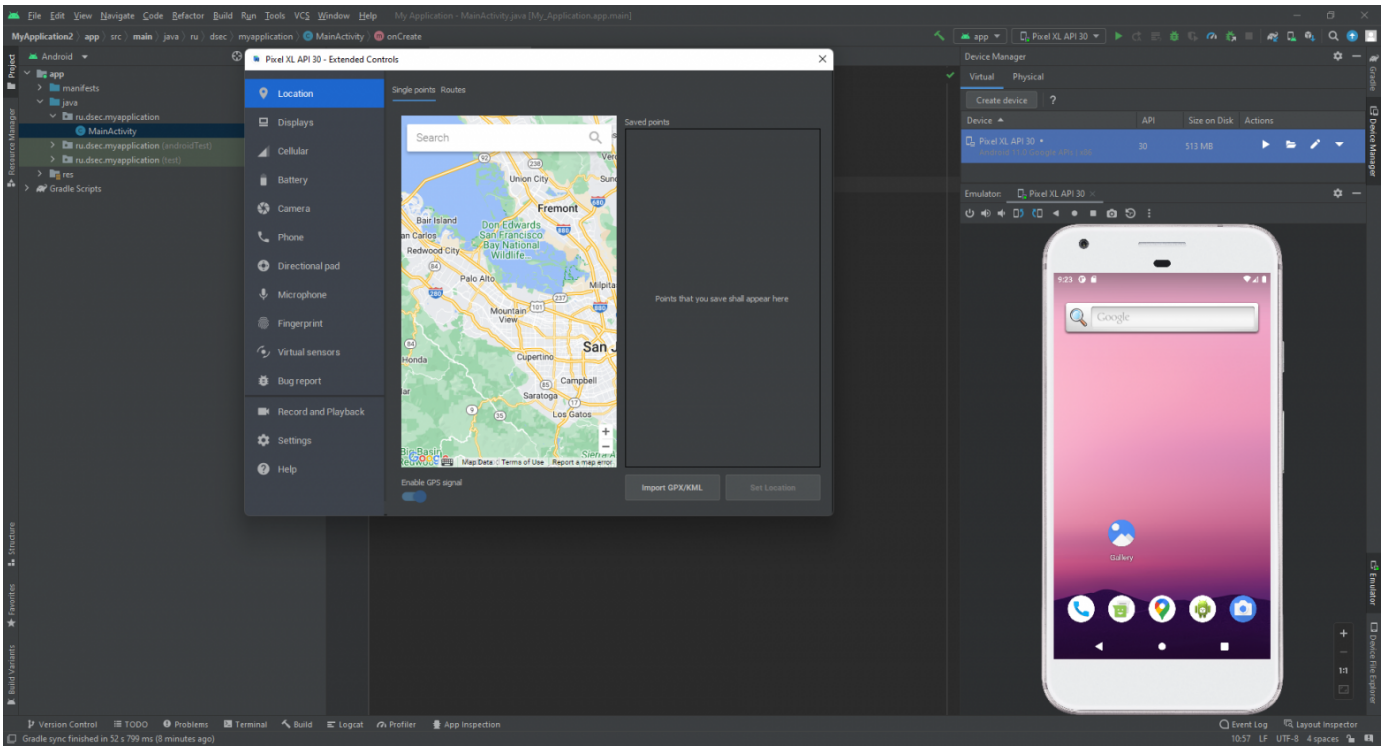
Android Studio Emulator

Чтобы создать виртуальное устройство, создаем пустой проект в Android Studio, выбираем вкладку Tools, где нам нужен инструмент Device Manager. Откроется боковая вкладка с девайсами.

На следующем шаге мы можем выбрать из двух версий — с Google Play и без. Если выбрать первую, то у вас не будет возможности выйти в режим с правами суперпользователя.



После скачивания необходимых файлов на вкладке Device Manager появится ваше устройство. Можем запускать.



Windows Subsystem for Android

Про этот интересный инструмент мы уже рассказали в прошлой статье по настройке WSA — обязательно посмотрите :)

Прокси

Анализ мобильных приложений неразрывно связан с анализом трафика между сервером и клиентом (мобильным приложением). Для этого можно найти множество (нет) инструментов. Но у нас в основном встречается три. О них ниже.

BurpSuite

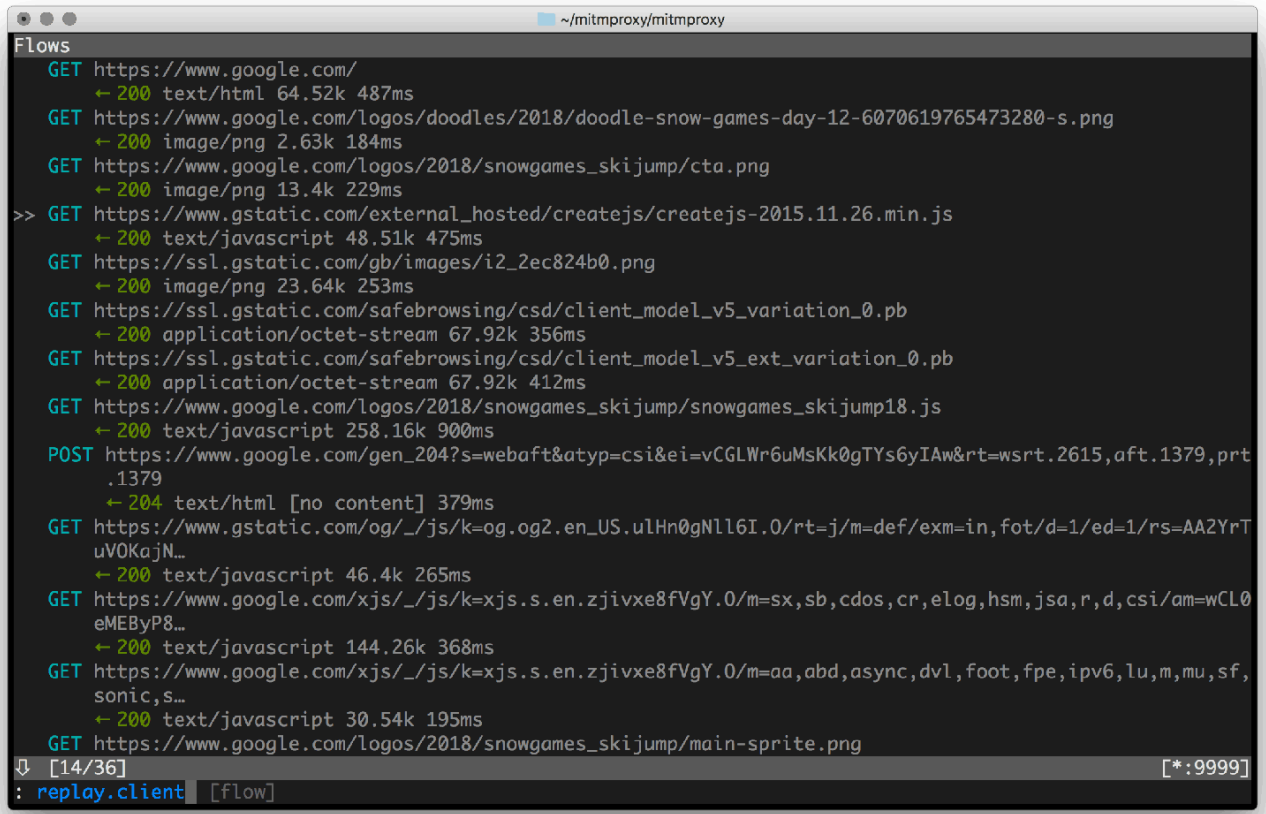
https://portswigger.net/burp/releases/

Это наш главный инструмент, который запущен почти всегда. Как бы это странно ни звучало, но у него самый понятный интерфейс и есть все, что нужно, из коробки. Распространяется в двух версиях — Community и Professional. Для базового анализа хватит и первой версии, к тому же всегда можно добавить нужные расширения через Extender.

MitmProxy

https://mitmproxy.org/

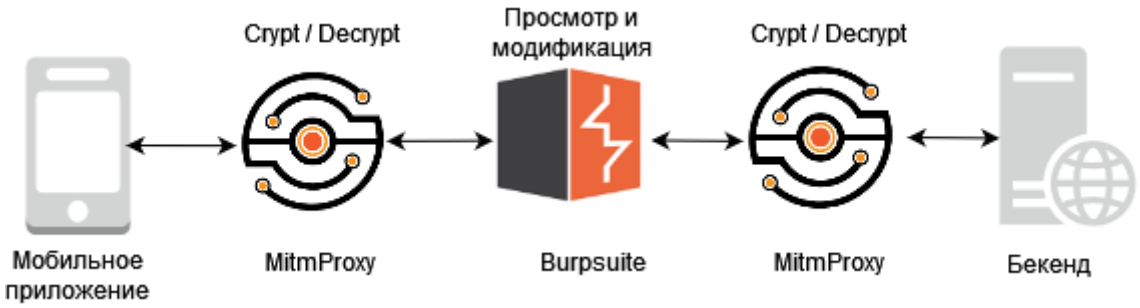
Консольная тулза, которая позволяет, как и Burp, завернуть в нее трафик и изучать его. Есть еще веб-интерфейс, но более интересна другая функциональность.



MitmProxy позволяет легко дописывать функционал при помощи Python API. Например, у нас был кейс, когда было реализовано шифрование данных, пересылаемых через вебсокеты, и хотелось посмотреть на них. А еще больше хотелось их модифицировать.

Буквально за несколько минут был написан скрипт, который расшифровывал данные (у нас были необходимые ключи), передавал их дальше в Burp, который был указан как upstream proxy, и был еще один upstream proxy, который зашифровывал данные и передавал уже на бэкенд.

Получился примерно такой флоу:



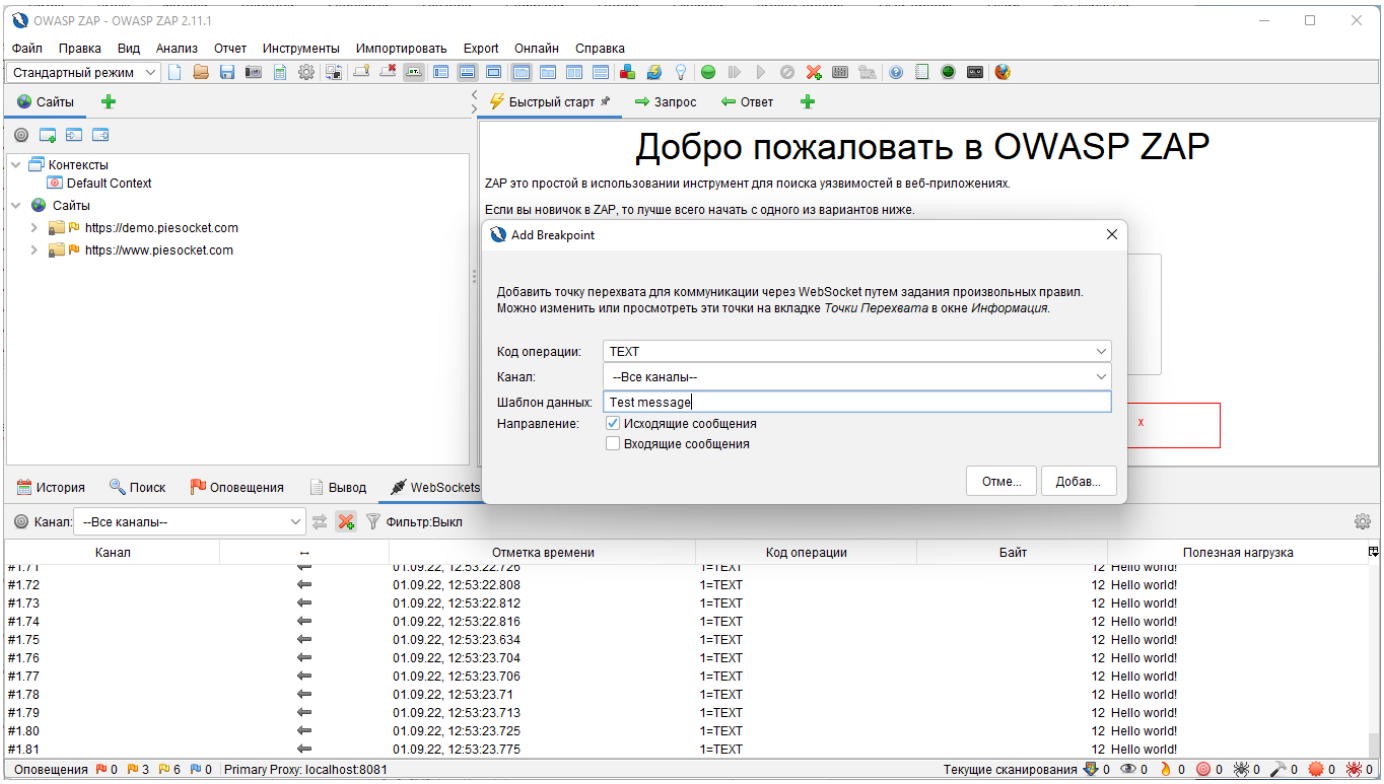
OWASP ZAP

https://www.zaproxy.org/download/

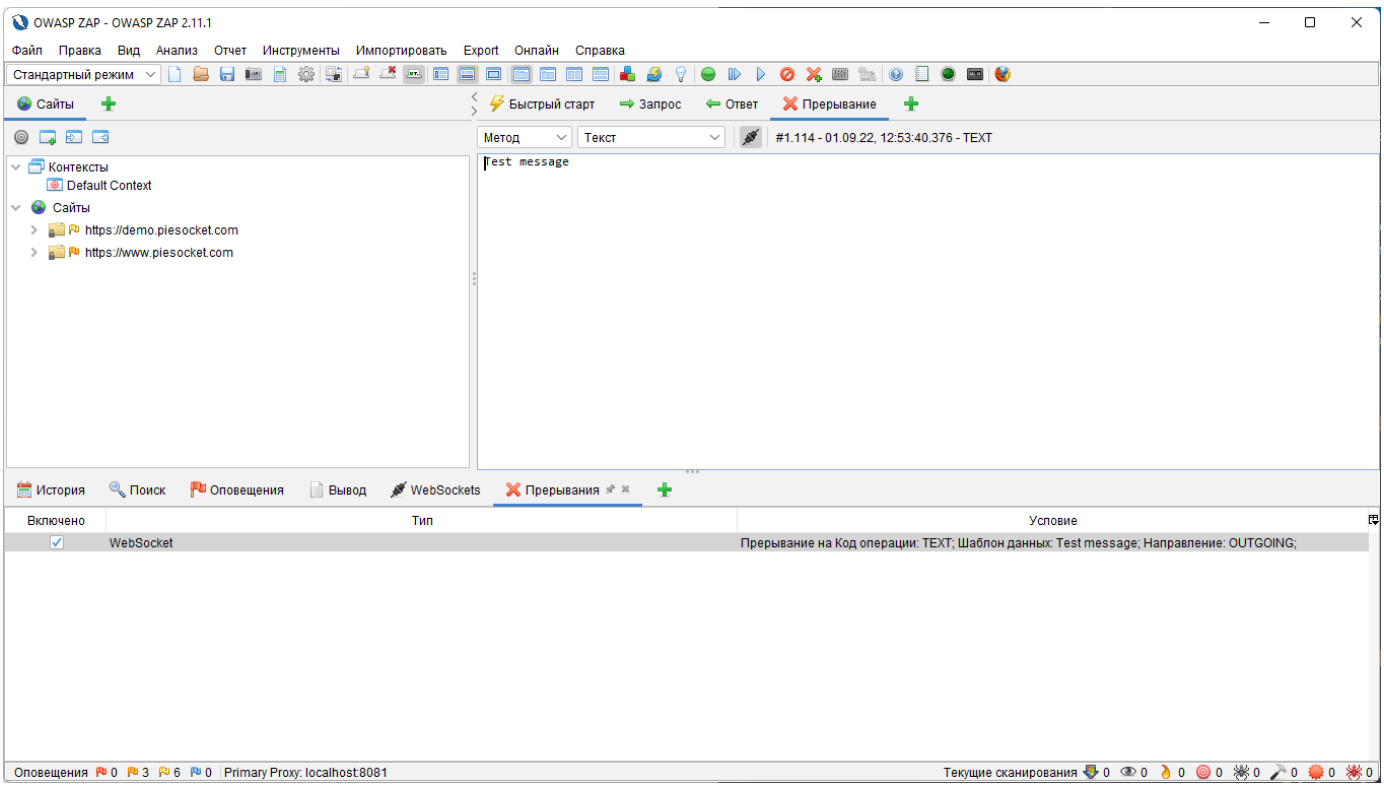
Еще один мощный инструмент, который практически не уступает BurpSuite. Единственный его минус — миллиард настроек, в которых можно бесконечно искать необходимую галку.

Хочется выделить одну из функций, которой нет в BurpSuite и которая при этом иногда необходима — возможность ставить точку остановки для вебсокетов. В современном мире встречаются сайты или приложения, у которых почти все клиент-серверное взаимодействие основано на вебсокетах. В BurpSuite вы можете включить Interceptor и прокликивать каждое сообщение, пока не появится то самое, нужное вам, которое вы хотите модифицировать и отправить дальше. Но что делать, если таких сообщений десятки или сотни? Здесь и пригодится этот функционал.

Добавить брейкпойнт можно на вкладке с вебсокетами. Мы определяем тип передаваемых данных и данные для поиска.



Как только появится сообщение, которое удовлетворяет нашему шаблону, произойдет прерывание, а ZAP покажет искомое сообщение.

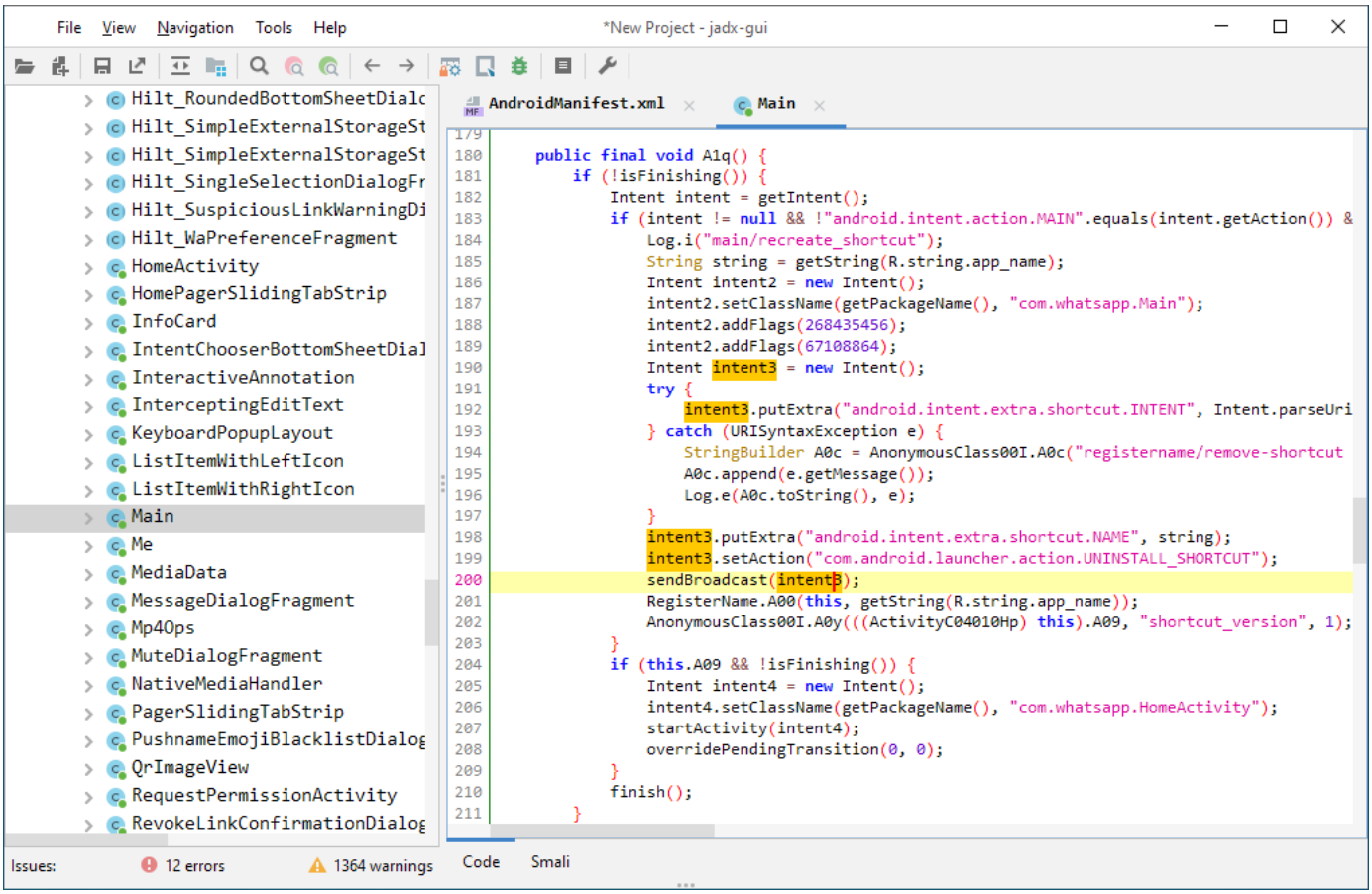


Статический анализ

JADX

https://github.com/skylot/jadx

Инструмент, с которым знакомы почти все аналитики безопасности мобильных приложений и не только. Это самый настоящий мастхэв при анализе приложений. Если при изучении трафика мы 70% времени проводим в BurpSuite, то при исследовании мобильного приложения «черным ящиком» 80% времени открыт JADX.



Позволяет на лету декомпилировать код, просматривать классы, ресурсы и делать поиск по всему приложению. Самый настоящий комбайн в хорошем смысле этого слова.

Также позволяет подключиться к приложению в режиме отладки и ставить брейкпойнты.

Apktool

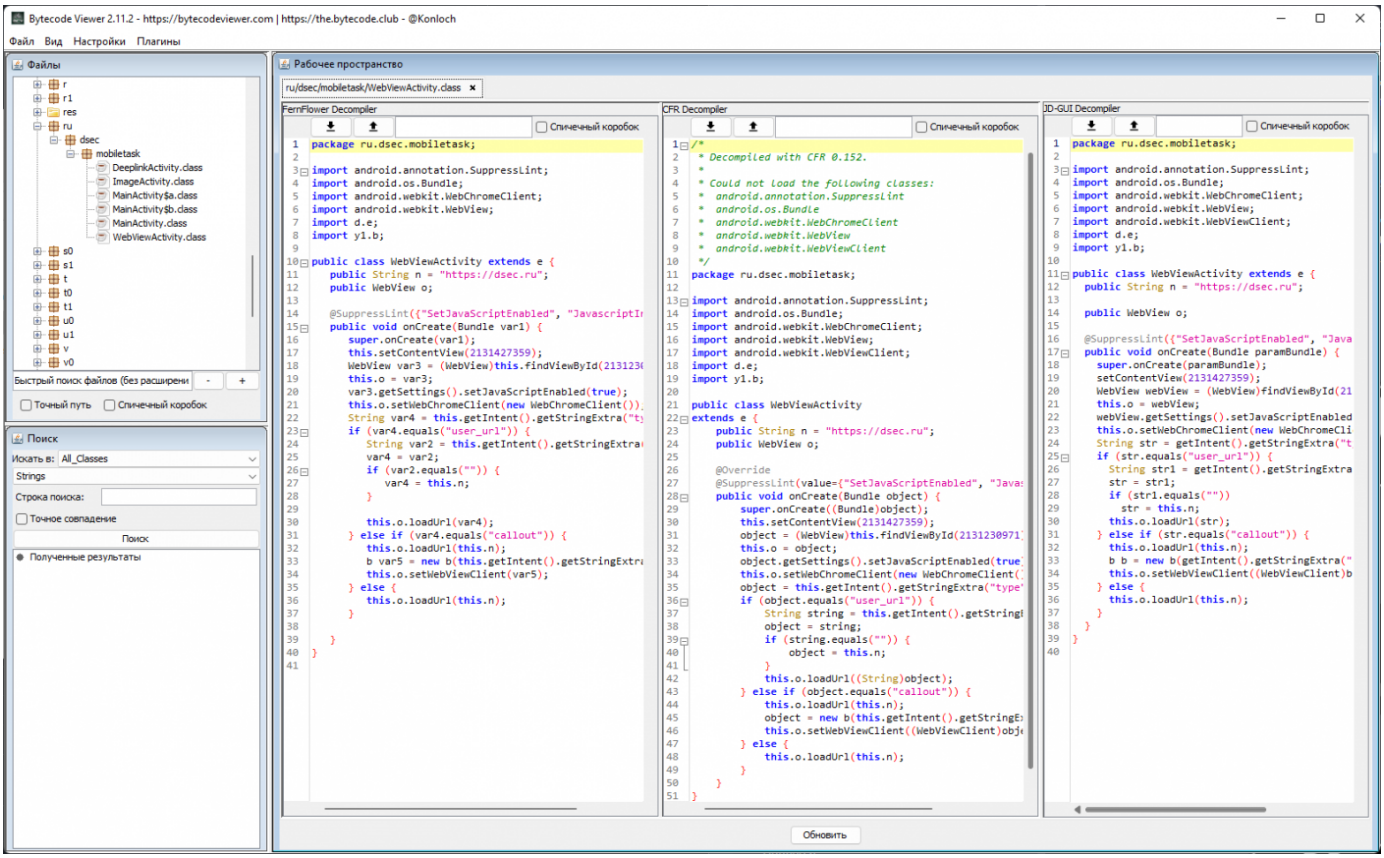
https://ibotpeaches.github.io/Apktool/

Используется ровно в двух случаях — когда нужно достать используемые ресурсы или ассеты, а также когда JADX не хватает памяти для того, чтобы открыть и декомпилировать все приложение, чтобы поискать по нему. В таком случае распаковываем приложение при помощи apktool, делаем поиск обычным grep/ripgrep и открываем класс либо в jadx, либо файл со smali-кодом в любом текстовом редакторе.

ByteCode Viewer

https://github.com/Konloch/bytecode-viewer

All-In-One тулза для анализа мобильных приложений. Можно закинуть APK и сравнить результаты сразу нескольких декомпиляторов. Очень полезно, когда JADX не может привести некоторые методы к нормальному виду, а через smali ничего не понятно :)

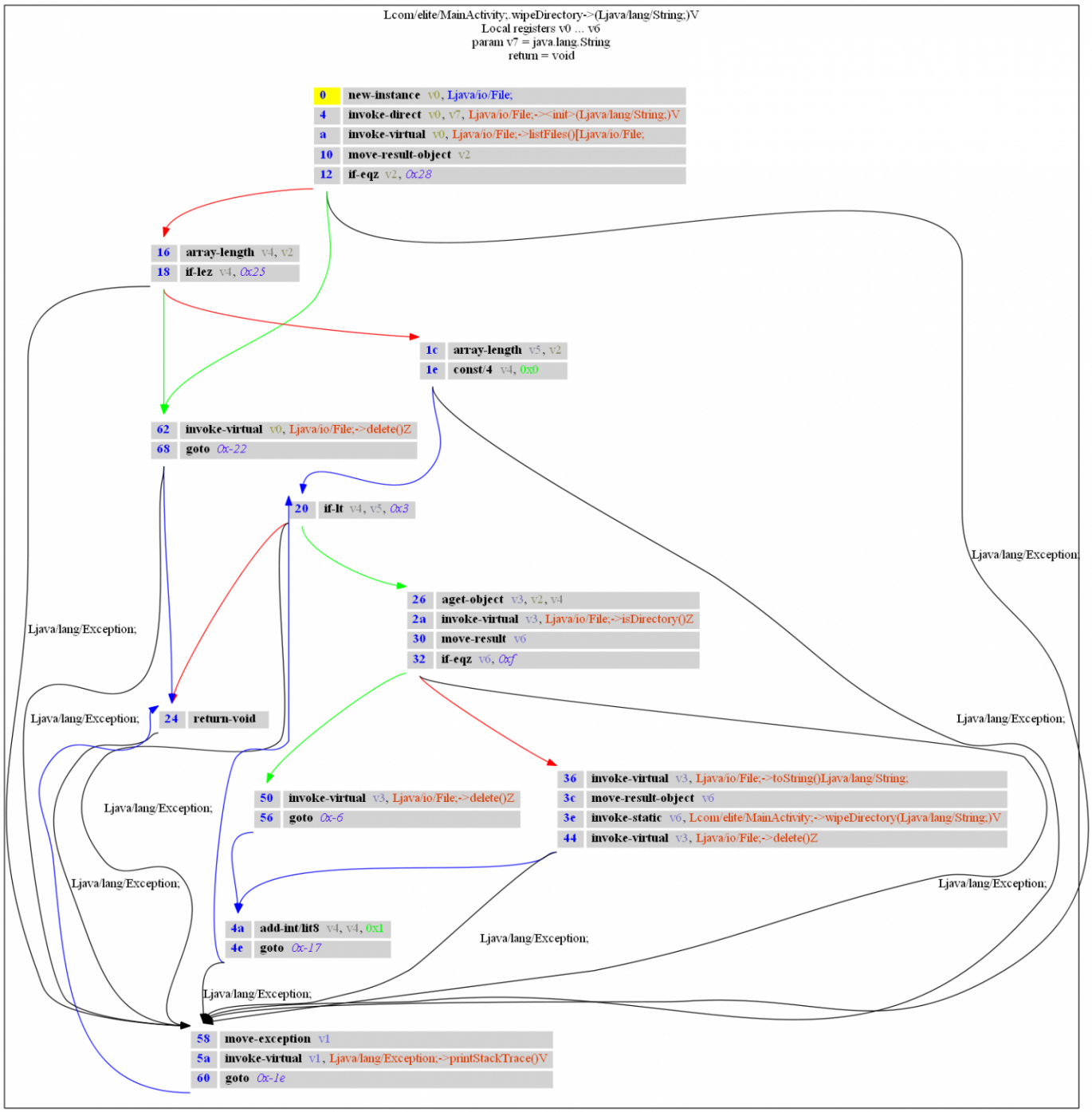


Androguard

https://androguard.readthedocs.io/en/latest/

Очень мощный инструмент для реверс-инжиниринга мобильных приложений. Несколько раз использовали его для автоматизации проверок, когда необходимо было пройтись по функциям и XREF'ам.

Также умеет рисовать графы для классов, что визуально может помочь разобраться в флоу работы мобильного приложения.



Mariana Trench

https://github.com/facebook/mariana-trench

Один из наиболее "свежих" инструментов для анализа мобильных приложений. Производит Taint-анализ по байткоду Dalvik. Легко устанавливается, из коробки уже имеет некоторый набор проверок, что поможет найти низко висящие уязвимости. Естественно, под каждый проект лучше писать свои правила — это повысит эффект от анализа и позволит сэкономить время.

Главное — учесть, что если у вас приложение большое и имеет кучу функционала, то нужно запастись временем и терпением — анализ может занять до получаса. Еще из замеченного: mariana trench не учитывает, экспортируемый ли компонент.

Представим базовую уязвимость, когда у нас в активити есть обработка диплинка, из него берется параметр `url` и передается в функцию `loadUrl` какой-нибудь webview. Mariana Trench подсветит это как уязвимость, однако если точка входа (Activity) у нас неэкспортируемая, то это будет False Positive.

Хоть это и стоит иметь в виду, в отчет такое не напишешь.

Hbctool

https://github.com/bongtrop/hbctool
https://suam.wtf/posts/react-native-application-static-analysis-en/

Консольная утилита для работы с байткодом Hermes. Одним из популярных фреймворков для кроссплатформенной разработки является React Native. Когда вы анализируете такое приложение, у вас есть три варианта развития событий:

- В файле `./assets/index.android.bundle` будет минифицированный javascript-код, который можно попробовать прогнать через beautifier и прочесть.
- Рядом с файлом `./assets/index.android.bundle` будет лежать файл `./assets/index.android.bundle.map` с маппингами для собранного файла, и вы можете при помощи https://github.com/rarecoil/unwebpack-sourcemap распаковать его и прочесть оригинальный исходный код.
- В файле `./assets/index.android.bundle` будет бинарщина. Это и есть Hermes Bytecode.

Эта тулза на крайний случай. При помощи нее можно попробовать превратить код в так называемый HASM. Читать его все еще трудно, но это явно лучше, чем бинарный файл. И через кровь и слезы можно попытаться изучить логику работы, изменить и собрать

обратно.

Динамический анализ

Frida

https://frida.re/

Самый известный инструмент, который позволяет на лету подключаться к приложению и внедрять в него свой код. Если сравнивать с чем-то по назначению, то в голову приходит только Xposed Framework. Если сравнивать по времени, то написание скрипта для Frida занимает намного меньше времени, чем модуля для Xposed.

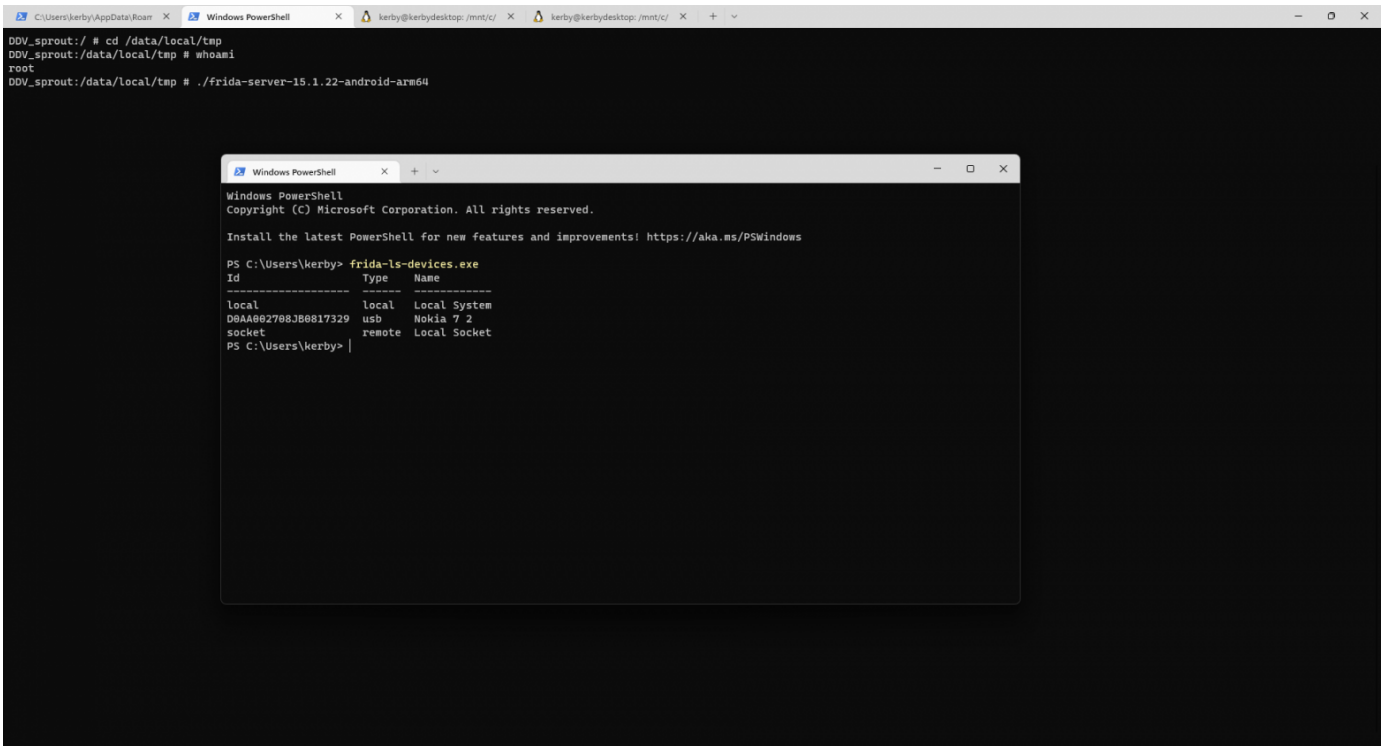
Чтобы установить необходимые вещи на хост, нужно выполнить следующую простую команду:

```
pip install frida-tools
```

Скачиваем frida-server для нужной архитектуры, переносим его на устройство и запускаем с правами суперпользователя:

```
su
cd /data/local/tmp
/data/local/tmp/frida-server-15.2.2-android-arm64
```

После этого мы сможем увидеть его в консоли и начать писать скрипты для приложений.



Objection

https://github.com/sensepost/objection

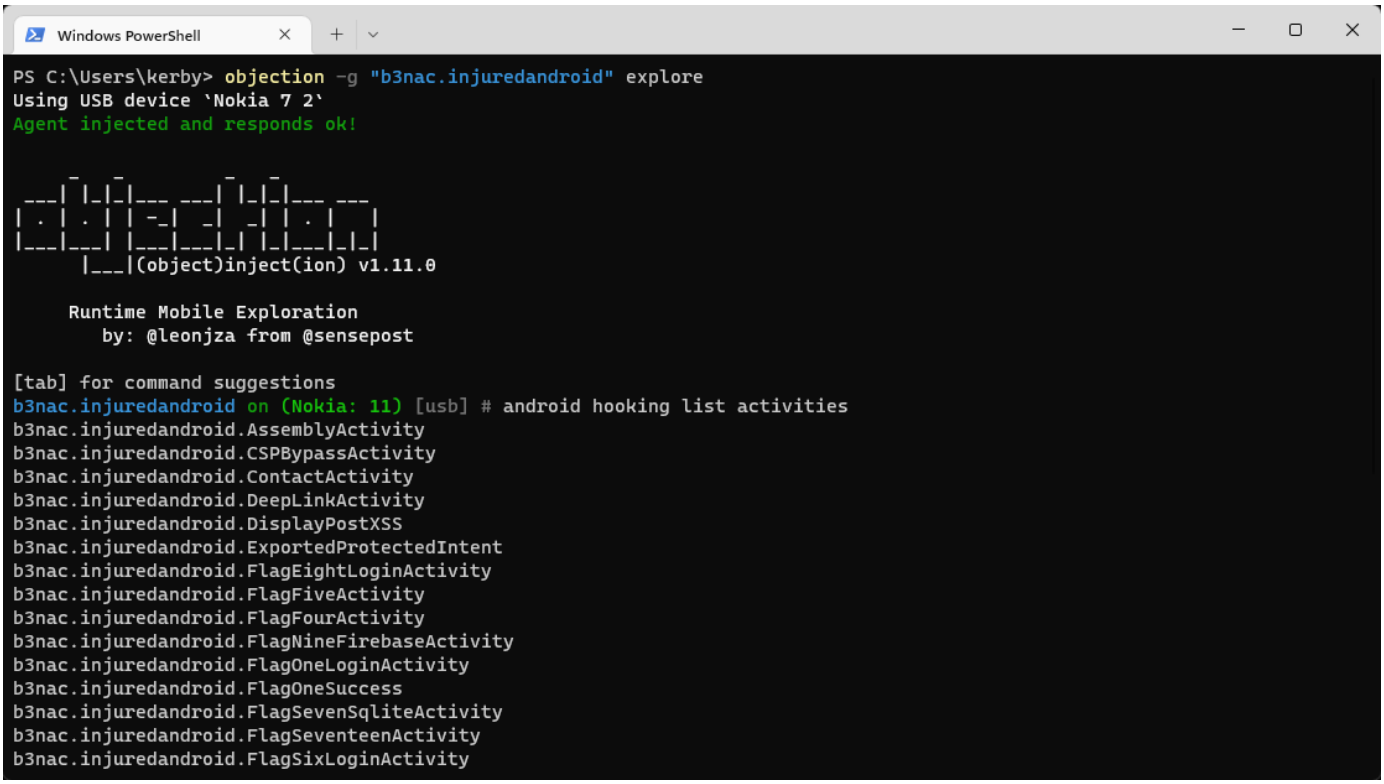
Objection — это тулкит, который основан на Frida и предоставляет уже готовые функции для работы с мобильными приложениями. Имеет следующие готовые функции:

- Работа с файловой системой

- Байпасс SSL-pinning
- Дамп keychain & keystore
- Работа с памятью
- Манипуляции с интентами

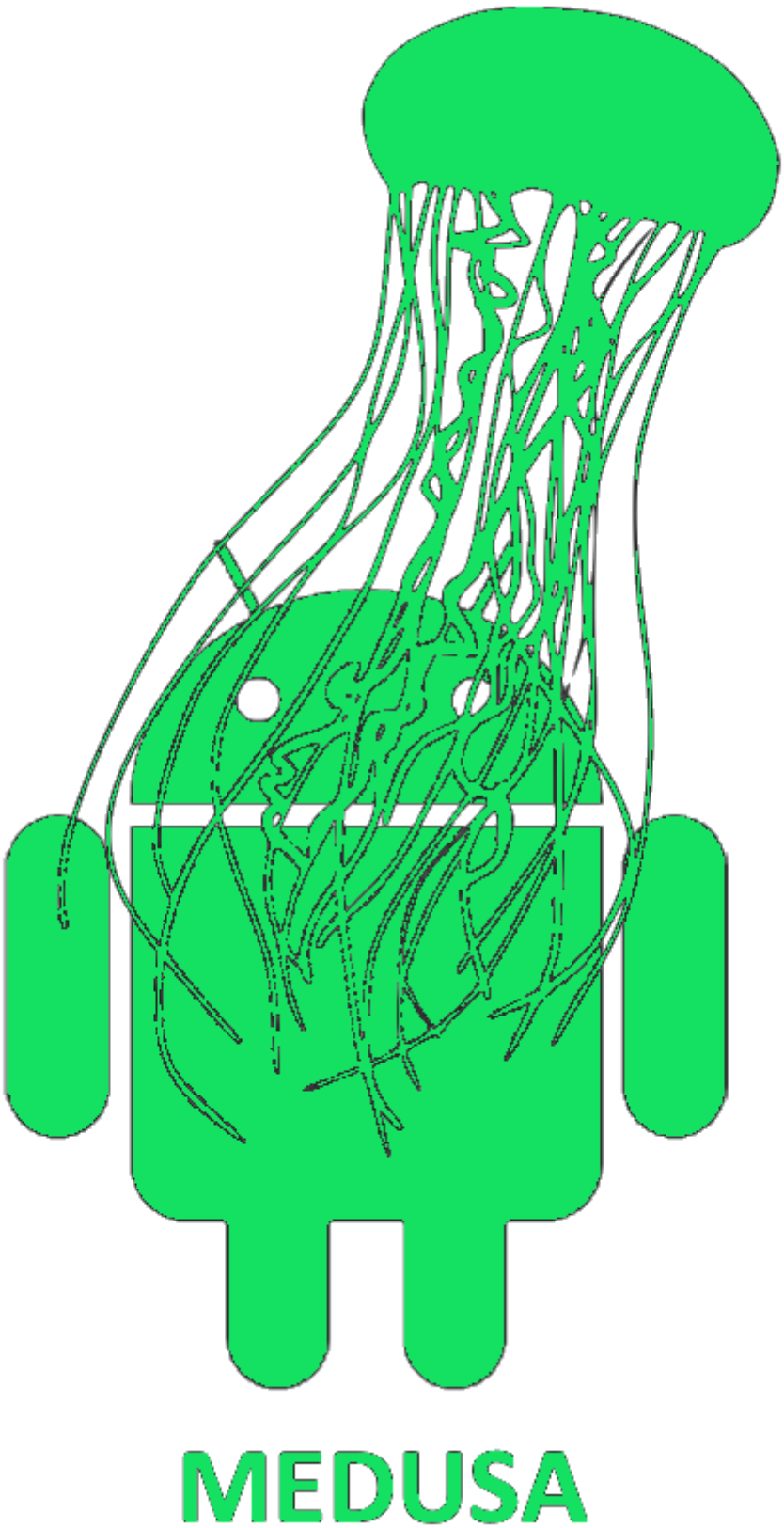
Установка: `pip3 install objection`

Запуск: `objection –g "package_name" explore`



А еще у него просто шикарнейшие автоподсказки.

Medusa



В моем представлении, это андерграунд среди средств для анализа мобильных приложений. Он мало где упоминается, что очень зря. Это самая большая коллекция скриптов для Frida из тех, что я видел. Можно сравнить с metasploit по подходу: у вас есть коллекция скриптов, вы выбираете необходимые, они компилируются вместе и инжектятся в процесс.

Очень удобно и иногда экономит кучу времени. Однако пару раз в модулях находились опечатки и ошибки, из-за чего скрипты ломались, так что приходилось править руками :)

Magisk-модули

Если посмотреть на Magisk-модули, то не так уж и много необходимых нам, которые позволяют сэкономить время.

MagiskTrustUserCerts

https://github.com/NVISOsecurity/MagiskTrustUserCerts

Таким модулем является MagiskTrustUserCerts. Современные версии Android не дают возможности добавить системный CA. Это надо делать вручную или при помощи таких модулей, которые на этапе загрузки системы перенесут клиентские сертификаты в системный каталог. Это необходимо, чтобы приложения доверяли нам, когда мы заворачиваем трафик на прокси для последующего анализа.

Данный модуль уже упоминался нами, когда мы настраивали WSA.

XPosed-модули

Ввиду устаревания оригинального XPosed, мы используем его форк — LSposed. Данный форк — это модуль для Magisk, из-за чего установка производится буквально в несколько кликов.

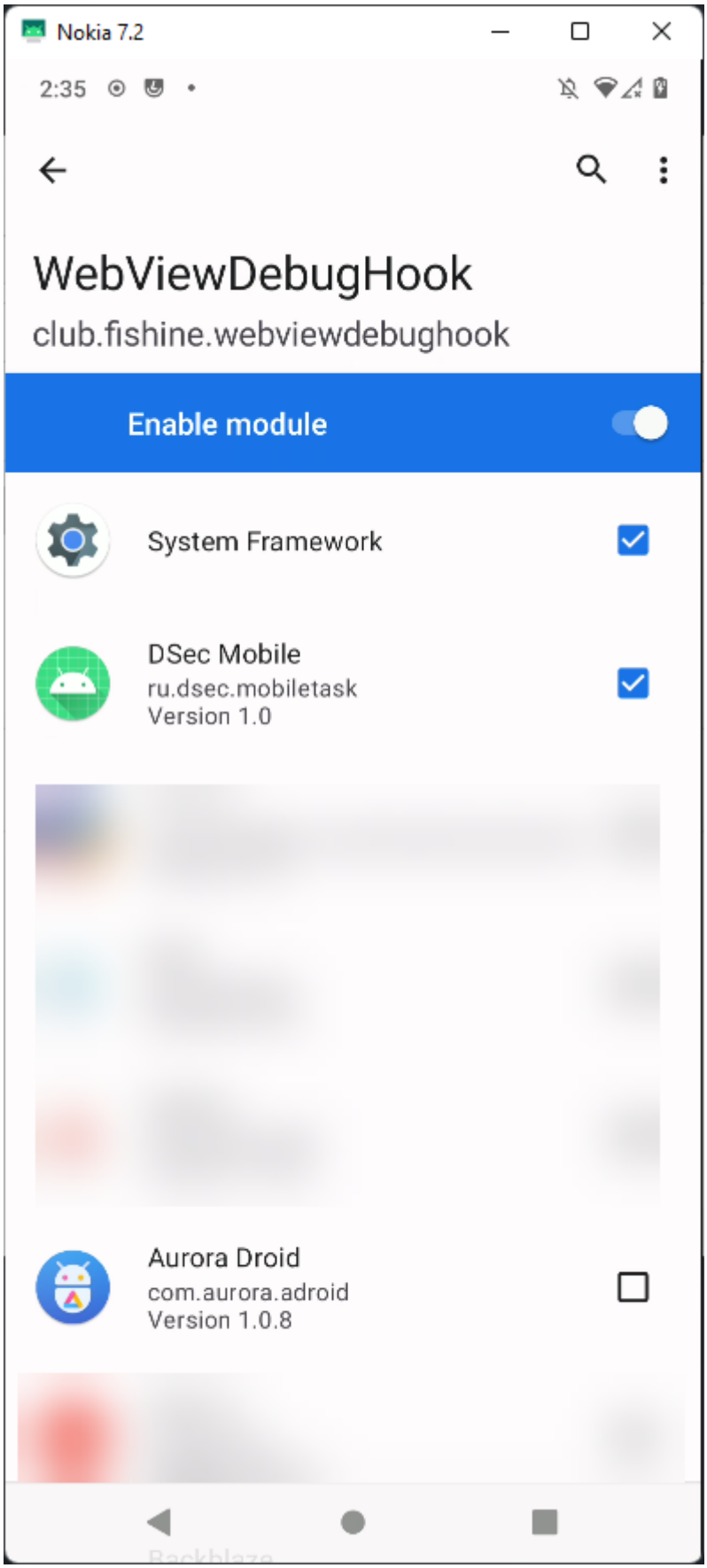
WebViewDebugHook

https://github.com/feix760/WebViewDebugHook

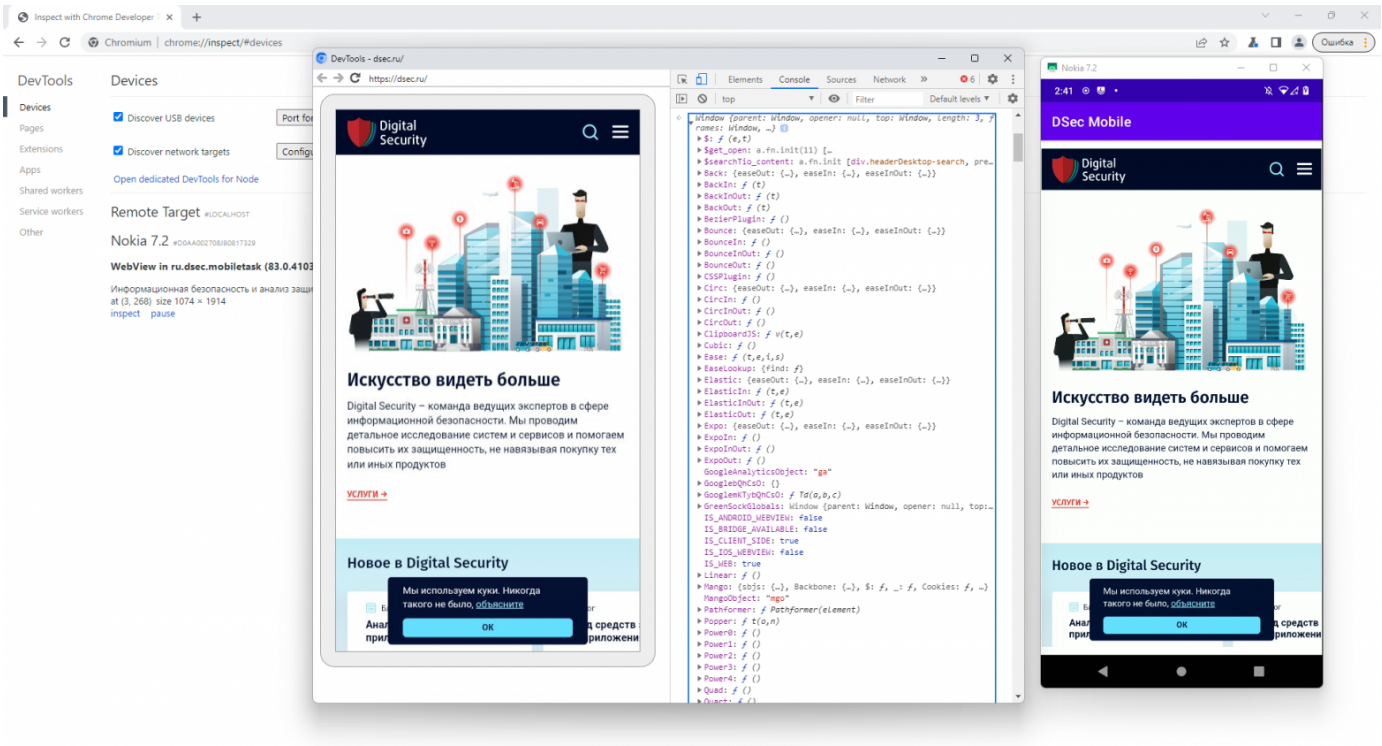
Мало что можно написать про этот модуль, но он делает буквально следующее:

```
WebView.setWebContentsDebuggingEnabled(true);
```

Удобно, если приложение использует WebView и нужно взглянуть на нее "изнутри". Устанавливаем данный модуль, через LSposed выбираем скоуп приложений, на которые будет распространяться данный модуль.

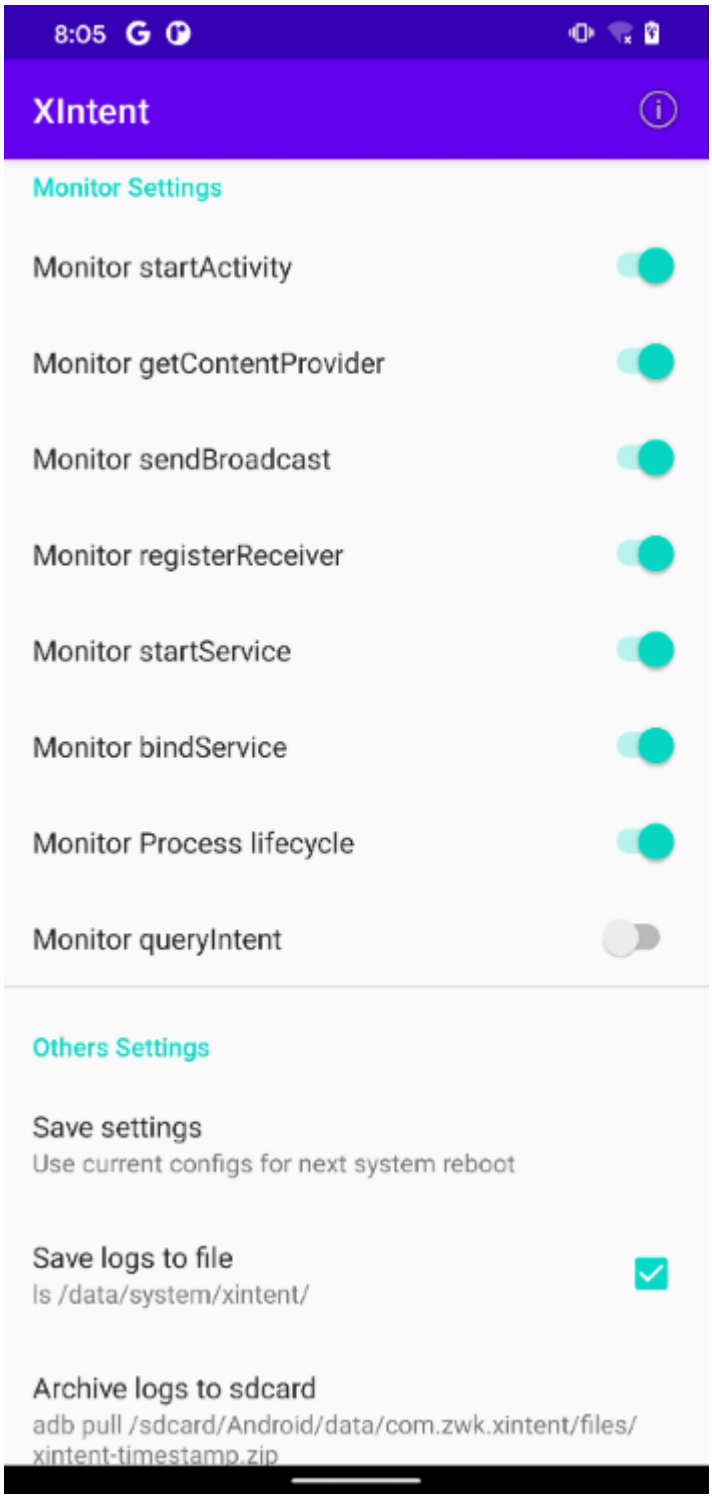


Теперь, когда у нас на активном экране будет WebView, мы сможем посмотреть на нее через Chrome (`chrome://inspect`).

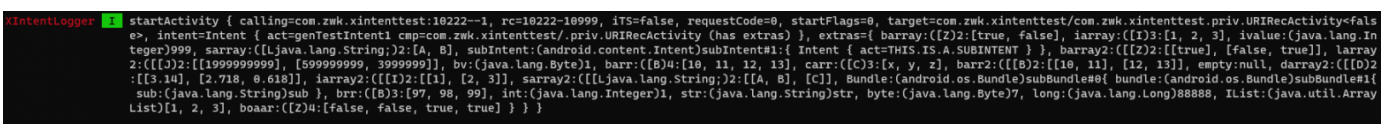


Xintent

Данный модуль позволяет видеть работу системы с интентами, которыми обмениваются приложения или компоненты приложения. Если вы хотите понять лучше логику работы приложения в динамике, то это отличный вариант.



Как и раньше, нужно выбрать целевое приложение — и вперед смотреть логи.



ADB

Наверное, странно видеть тут ADB, но не упомянуть его нельзя. Он умеет многое и

играет роль чуть ли не основного инструмента при взаимодействии с устройством.

Проброс портов

```
# пробрасываем 7777 порт на устройстве, чтобы трафик шел на 8080 порт хоста
adb reverse tcp:7777 tcp:8080
```

Манипуляции с настройками

Командами ниже можно указать HTTP proxy и убрать его соответственно.

```
adb shell settings put global http_proxy 127.0.0.1:7777
adb shell settings put global http_proxy :0
```

Текущая on-top activity и fragment

```
adb shell "dumpsys activity activities | grep mResumedActivity"
```

Вывести список всех приложений и путь до base APK

```
adb shell pm list packages -f
```

ADB logcat по package name

```
linux:
adb logcat --pid=`adb shell pidof -s com.example.app`

windows:
adb logcat --pid=$(adb shell pidof -s ru.dsec.mobiletask)
```

Запустить intent

Базовый пример:

```
adb shell am start -a android.intent.action.VIEW -d "dsec://open"
```

```
adb shell am start -a android.intent.action.VIEW -d "dsec://open"
```

`am` умеет практически все. У него длинный, но крайне понятный manual :)

Прочее

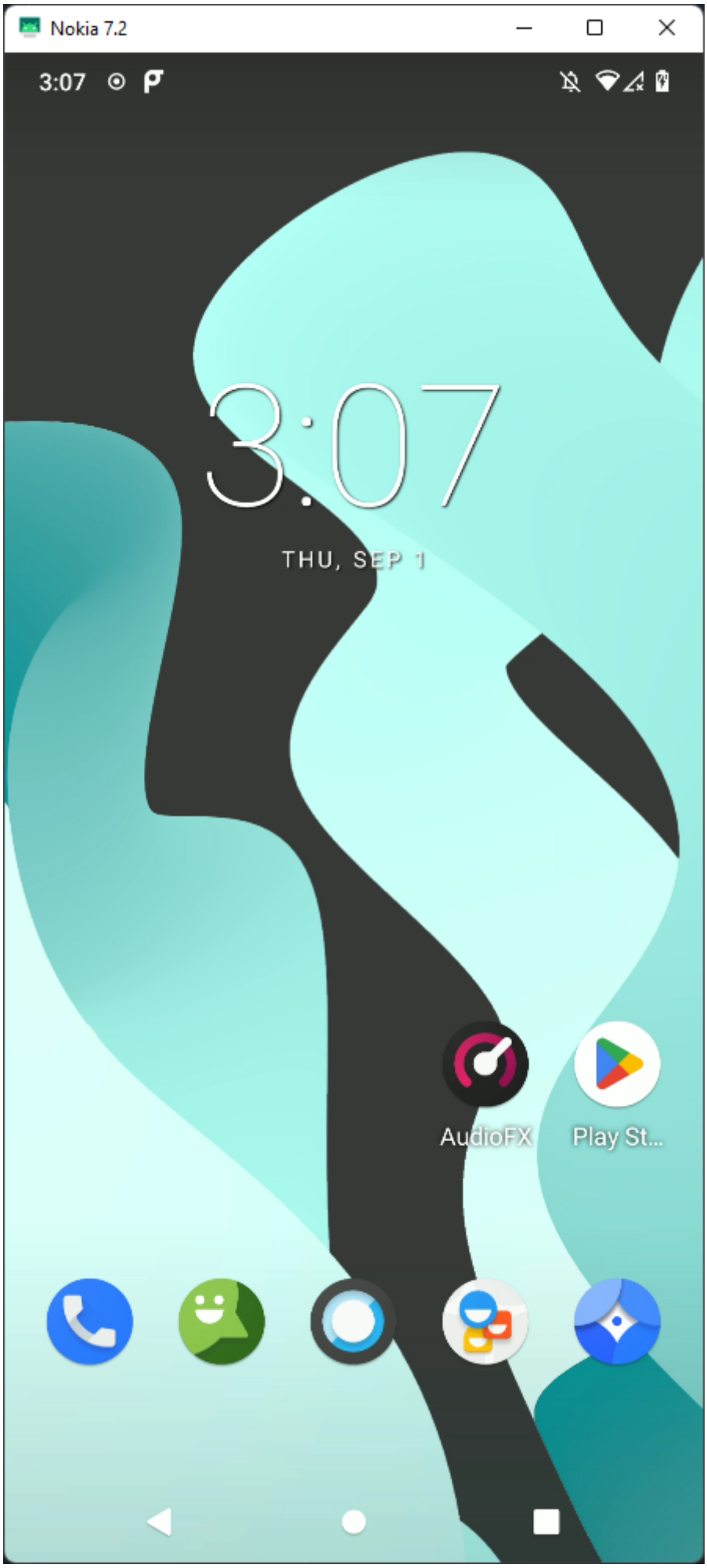
Сюда попало все, что не вошло в остальные категории.

Scrcpy

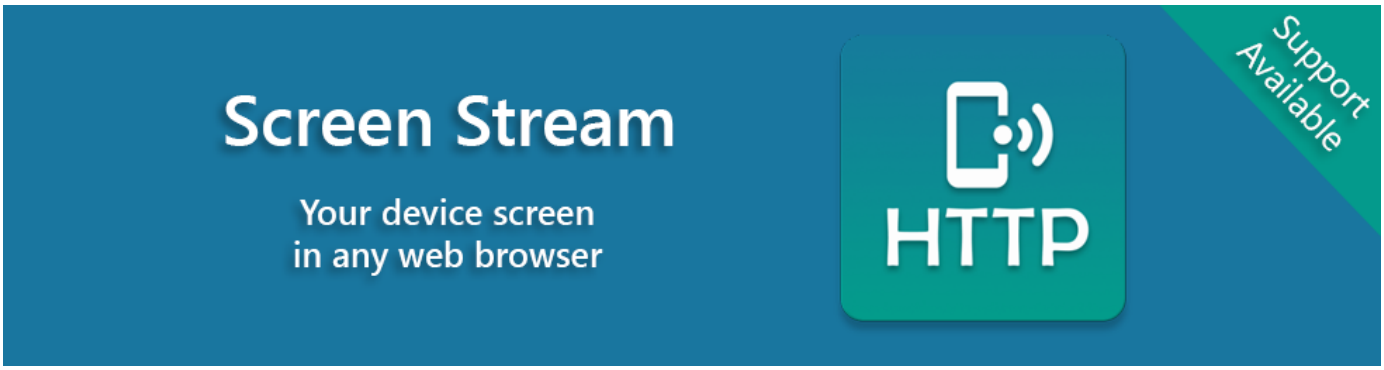
https://github.com/Genymobile/scrcpy

Это приложение позволяет отобразить экран Android-девайса, доступного через ADB. Также позволяет вводить данные через клавиатуру. Очень удобно, когда нужно вводить огромный пароль от тестовой УЗ и делать это через телефон крайне лениво :)

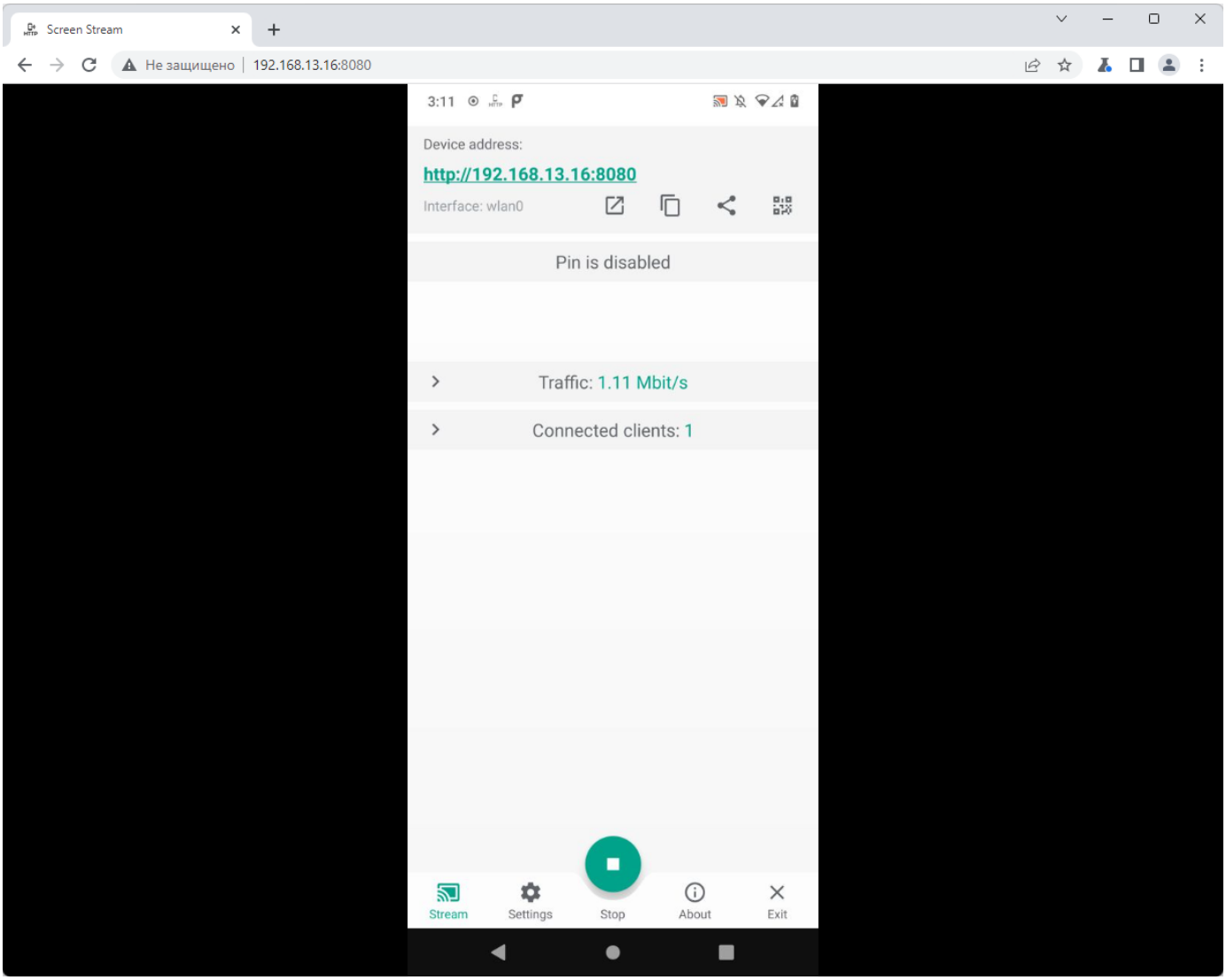
Есть под все ОС, что довольно редко для наших дней.


Nokia 7.2
3:07
3:07
THU, SEP 1
AudioFX
Play St...

ScreenStream



Казалось бы, зачем он нужен, когда есть scrcpy. Но что если нет возможности подключиться по проводу и вы находитесь в одной сети? Данное приложение нужно не очень часто, но помнить о нем полезно.



Работает с полпинка, главное — иметь сетевой доступ.

Заключение

В этой статье представлены не все, но большинство инструментов, которые мы используем каждый день при работе с мобильными приложениями. Надеемся, что это поможет людям, которые только погружаются в анализ мобильных приложений и хотят собрать первое рабочее окружение.

Есть что дополнить? Пишите в комментарии.

Теги: android, pentest, инфобез, аудит безопасности, статический анализ, динамический анализ, мобильные приложения

Хабы: Блог компании Digital Security, Информационная безопасность, Разработка под Android, Тестирование мобильных приложений

Digital Security
Безопасность как искусство

Сайт Facebook Twitter Github Telegram ВКонтакте

17 Карма 27 Рейтинг

kerby @kerbyj
Пользователь

Комментарии 3

drhouse
13.09.2022 в 08:23

Однозначно в закладки! Спасибо!

+2 Ответить

whoam1ns3
22.09.2022 в 23:33

Там оно и останется навсегда

0 Ответить

whoam1ns3
22.09.2022 в 23:33

А почему нет genymotion? Меньшее потребление ресурсов, рут из коробки...

0 Ответить

Только полноправные пользователи могут оставлять комментарии. Войдите, пожалуйста.

ПОХОЖИЕ ПУБЛИКАЦИИ

29 июля в 16:54

Анализ iOS-приложений

+7 2.4K 26 2 +2

19 ноября 2018 в 05:00

Клонируем бесконтактную карту с помощью мобильного приложения

+42 131K 256 21 +21

2 августа 2017 в 11:04

Внедряем безопасность в процесс разработки крупного проекта

+40 21K 145 6 +6

ЛУЧШИЕ ПУБЛИКАЦИИ ЗА СУТКИ

сегодня в 01:20

Ошибки выбора MongoDB в качестве основной БД в стартапе

+66 13K 55 55 +55

сегодня в 13:23

Кейс: Как ошибка в рекламе принесла прибыль в 600 000 рублей

+49 6.4K 7 14 +14

вчера в 20:19

Когда понты дороже денег: Оживляем Java подделку iPhone 4s, и смотрим на что она способна

+34 11K 18 20 +20

вчера в 18:37

Брюнетки против блондинок или как на мониторе показать цвет свечения светильника?

+33 3K 22 9 +9

сегодня в 02:20

Ниже некуда? Продажи ПК и ноутбуков падают гораздо быстрее прогнозных значений

+28 10K 12 32 +32

Вернуться на старую версию

ИНФОРМАЦИЯ

Дата основания	16 апреля 2003
Местоположение	Россия
Сайт	dsec.ru
Численность	51–100 человек
Дата регистрации	23 марта 2012

ССЫЛКИ

Сайт Digital Security
dsec.ru

Digital Security в Telegram
t.me

Digital Security ВК
vk.com

Digital Security на YouTube
www.youtube.com

Блог Digital Security на VC
vc.ru

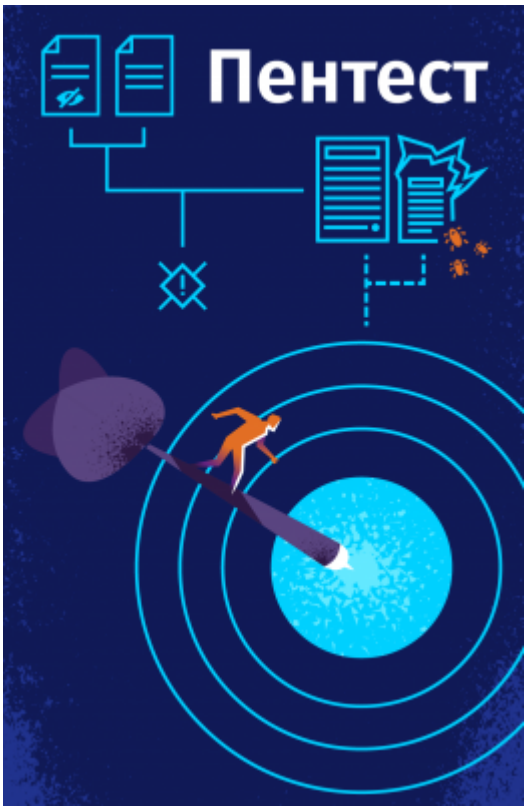
Digital Security на Дзене

Digital Security на Дзене
dzen.ru

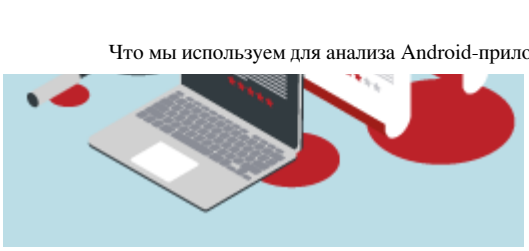
Международная конференция ZeroNights
zeronights.ru

HeadHunter Digital Security
spb.hh.ru

ВИДЖЕТ



ВИДЖЕТ



БЛОГ НА ХАБРЕ

вчера в 05:00

Полевой набор пентестера

8K 15 +15

30 сентября в 09:55

Фаззинг JS-движков с помощью Fuzzilli

1.7K 2 +2

13 сентября в 07:01

Что мы используем для анализа Android-приложений

3.8K 3

7 сентября в 07:00

Готовим Android к пентесту — WSA edition

2.8K 3 +3

31 августа в 04:00

Атакуем синезубого короля

7.2K 22 +22