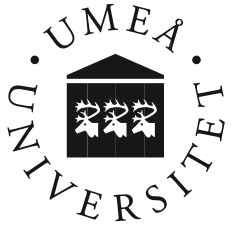

Android Security

An evaluation of applications in Google Play

Jonas Torstensson

Jonas
Torstensson

VT 2017

Bachelor Thesis, 15 hp
Supervisor: Pedher Johansson
Examiner: Kai-Florian Richter
Bachelor of Science Programme in Computing Science, 180 ETCS

Abstract

This thesis examines security issues that might occur in the applications from Google Play. It examines vulnerabilities by an evaluation of how well the policies and guidelines of the store transfers to the applications.

After a short literature study about android and security. The polices and guidelines of Google Play is examined. Experiments with 10 of Google Plays top listed applications are conducted. Then a simple pen test explores how this may affect the security in the user's phones. Lastly the result is discussed in respect to guide lines and policies and Google Play.

CONTENT

1 INTRODUCTION

Over a period of 10 to 15 years the market for smartphones has rapidly grown into one of the largest in the computer/electronic industry. According to Statista 2.32 billion people own a smartphone today [1].

It's constantly growing list of usages which by now has become both long and varied puts the smart phone in the centre of many people's digital life. Teetering it together like the spider in their private home web handing out services and transferring information. There's no doubt this development will continue and with bank applications, door locks, digital identification, internet connection and internet of things emerges a growing need for security [2, 3].

The smart phone developers have met this need by redirecting their users to their own stores, implementing encryption and by limit the users access to their phones by prohibit root access [4, 5, 6]. But as much as everyone wants security, for the ordinary user flexibility and effectiveness often out weights it [7].

The conflict can be seen in the battle between the two leading brands of operating systems for smartphones today Android and IOS which Android seem to be winning with 87.5% of the market shares [8].

Android with its open code and permissive nature gives opportunity for the community to help to enrich the platform. but it also gives opportunity to write bad malicious code [9]. This has given many of the app stores for Android a bad reputation, and Google is trying to limit its users to its own store Google Play to give them a secure environment to download apps in [10, 11].

But even if Google Play has less malware compared to many of the other app stores on the internet, it too has over and over been the target for attacks and the list of malicious apps found in the store is long and growing [12, 13].

1.1 Background

Google has made many attempts to clean Google Play store from malicious applications. But that may be easier said than done with over 2.8 million applications on the site and several 1000 added every week [14].

Many projects and papers about automatic vetting have been released since the first reports about malware on the Google Play store (android market) and many people have been involved in trying to make the store a safer place [15, 16]. Google Plays bouncer was just a couple of years after its release deemed to be insufficient by many security experts [17, 18] and now the applications go through both static and dynamic analysis before they are published. According to the Android Security 2016 Review, Google now review all new applications by performing a cloud based analysis before they become available in the store. The report claims that fewer than 0.71% android owners have potential harmful applications installed and if exclusively downloading from Google Play the number was even smaller 0.05% [19, 20].

But the problem with Google Play isn't just malicious apps and viruses, many of the applications written with good intentions have problems, and the security guidelines don't always seem to be the highest priority.

One example of this is the “Airdroid attack” 2016 where hackers were able to use a flaw (insufficient encryption) in the well-known application with about 50 million users to execute malicious code and pull sensitive data by sending SMS’s to the target device.

> “The Air Droid attack flow provides cybercriminals with a very easy way to target users: sending a contact card and an SMS message to execute the attack,” said Oded Vanunu, security research group manager at Check Point. “The main threat is a complete theft of private information – imagine, for example, that just receiving an SMS message can result in all of the user’s data being stolen. Another threat is that an attacker could control the content of the target’s device.” [21]

In 2014 Google Play started their App Security Improvement Program and began to put pressure on the developers to correct certain known security issues and have by now about 25 different issues on their remediation list [22, 23].

But still most of the android developer’s guidelines and core qualities is just partially enforced by Google Play and many of the application is as consequence flawed. This of course makes it much easier for an attacker to find some way of taking advantage of weaknesses in the code [21, 24, 25].

This thesis will examine what kind of security issues the applications from Google Play may have. By testing them against limited set of Google Plays policies and common guidelines. And then discuss how well they measure up to make the store a safe place for the users.

1.2 Problem description

What kind of security issues do applications from Google Play have? How do the applications measure up to Google Plays policies and the common guidelines?

1.2.1 Delimitation

To be able to finish the project in time it will be delimited to just two of the most common security issues on OWASPs.

1. Focus only on
    a. OWASPs first two TOP 10 issues:
        i. Improper platform usage
        ii. Insecure data storage
    b. Only client base issues
2. Examine 20 applications regarding the core app quality guidelines
    a. Top Google Play
        i. Random from the 540 most popular apps
    b. 10 randomized with at least 10000 downloads
    if any time left

[26, 27, 28]

2 ANDROID

2.1 Introduction

Android was developed under Google by Open Handset Alliance and is an opensource Linux based OS for mobile devices like smartphones and tablets [29]. The operating

system was released in September 2008 and has since become the dominating OS in the mobile market. Four years after release it had 25.5% of the market and by 2016 it had 87.5% [30, 31]. Androids success might have several reasons but the main reason is probably its permissive nature, openness and flexibility allowing it to adapt to new hardware and giving a wider range of options for third parties [32, 9].

2.2 Architecture

The Android architecture can be depicted as a software stack consisting of four conceptual layers see Figure 1.

Figure 1 Androids software stack and its conceptual layers [33]

The application layer on top of the stack consists of both native and third-party applications. They are tied together by the application framework layer in which different sets of content providers, services and managers help to let them do their work by giving them access to the rest of the system and by manage the lifecycle. In between the Linux kernel the C libraries provide the primary APIs to the Android runtime environment [34].

2.3 Linux kernel

The Android operating system is built with the Linux kernel as base. The Linux kernel has been used for many years and is known to be stable and secure. By being a multi user operating system it can provide a user-based permission model Users id and id groups are essential to the security in the Linux kernel. When the applications are installed they get

a user id applied to them. Permissions are set for every kind of resource in the Linux system. The permissions are divided over three categories

- Owner
- Group
- World (public global)

Mapping the user id to a category gives them that categories permissions to read, write or execute the file [35].

2.4 Applications

By using the Linux kernel's inbuilt multi user ability Android applications runs in a sandboxed environment isolated from each other and the rest of system. The applications can't reach outside its own memory sphere without using the kernel [36]. Thus, it is possible to restrict the applications access to other resources like networking, Bluetooth features and components in other applications. All applications consist of four basic components.

- Activity

  An object representing a single screen used by the user to interact with the application.
  https://developer.android.com/reference/android/app/Activity.html

- Service

  Background tasks doing some service, isn't visible to the user and runs independent from the applications using it.

- Content Provider

  Is used by the application to share data with the rest of the system. It open-ups an interface to the database giving other applications access to it allowing them to read or write by using androids SQLite database system

- Broadcast Receiver

  Used to receive intents (messages system of androids used for IPC) from the system and act upon the action it requests [37].

These components are by default isolated from the rest of the system but can by the settings in the manifest xml file be exported and by that be exposed to the rest of the system. The exposure can be limited by permissions, if a component has no permission it will be public to the entire system. This breaks the isolation, creates vulnerability's and should always be avoided, if possible the exported tag should be set to false to make them invisible to the system [38, 39].

The applications can have distinct types, basically there are three types of mobile applications

- Web
  Runs in device web browser developed with html5 or JavaScript to provide interaction with a backend sever
- Native

Run locally on the phone

- Hybrid
  Use html and JavaScript locally inside a native container
  [40]

2.5 IPC

To IPC with exported components in other applications intents are used. An intent is a message object that can be used to interact with other applications through actions. An Action exposes a certain method inside the component able to perform some sort of task in behalf of other applications. It is possible to build custom actions with the intent filters to express suitable generic actions (like view, or pick).

Intents can be either explicit and implicit meaning they can ask a specific component to do some action or ask the system for a component capable of preforming it. In case of multiple applications able to perform the action implicitly asked for, the user can decide which of them to use and set it as default [41].

2.6 Android Manifest

All applications have a manifest. The manifest contains of metadata which defines the structure of the application. The manifest is divided in elements. The elements define all the blocks inside the manifest by tags:

<type, attributes> //BLOCK OF CODE// </type, attribute>

These blocks declare all the applications parts and give them attributes depending on type of element. The attributes contain information about the element like name, permissions and how they should interact or what data to use.

The manifests top element is always of the manifest type. It contains attributes like version name package name, platform, minimum API etc... Inside the manifest element, the required (uses) permissions and features are first declared then the application element of the xml in which the core components of the application is declared see Figure 2, [38].

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.basiccontactables"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-permission android:name="android.permission.READ_CONTACTS"/>
    <!-- Min/target SDK versions (<uses-sdk>) managed by build.gradle -->
    <permission android:name="android"></permission>

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/Theme.Sample" >
        <activity
            android:name="com.example.android.basiccontactables.MainActivity"
            android:label="@string/app_name"
            android:launchMode="singleTop">
            <meta-data
                android:name="android.app.searchable"
                android:resource="@xml/searchable" />
            <intent-filter>
                <action android:name="android.intent.action.SEARCH" />
            </intent-filter>
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Figure 2 Example of an Android manifest from developer.android.com

[42]

2.6.1 Permission

A permission restricts the access to a resource on the device. If an application declares in its manifest it requires permission to a resource the user must grant it when installing the application (can be done run time on newer versions [43]). It is possible to set up custom permission to restrict access to the exported components of the application. All permissions requested by the applications are declared in the beginning of the android data block [36, 38].

2.6.2 Core Components

The core components of an application are all declared in its manifest. Elements of this type uses attributes like export and enable. The exported attribute decides if the component can be accessed from the outside. If the exported attribute is left out, the default value for the component will decide if it gets exported or not. For API > 16 all the components get the exported value set to false by default but for API<=16 the content provider is exported by default. All the exported values should because of this be set to false explicitly according to the core app quality test sheet [38, 28].

2.6.3 intent filters

A component can declare certain actions it can perform in the manifest by declaring an element of the type intent filter. An intent filter contains the name of the action, what type of data it uses and what kind of category the action belongs to [41].

Example of an intent filter:

```
<activity                                android:name="ShareActivity">
                                                        <intent-filter>
              <action   android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
                        <data      android:mimeType="text/plain"/>
                                                      </intent-filter>
</activity>
```

[38]

2.6.4 APK

AKP is an acronym for Android Application Package. The APK is the only file needed to install an application on an Android device. The APK file is just camouflaged traditional archive file which means it's possible to decompress it with an ordinary extraction tool (WinZip, WinRAR etc.). To do so is as simple as changing the name of the extension from APK to zip. In the uncompressed file, all the various parts of the APK file can be extracted. The APK file consists of:

- The classes in dex format
  The dex files is the dalvik bytecode generated from the source code. To decompile to java this file must be converted
- Resources
- Assets
  Video, music and such
- META-INF
  Certificates
- Res
- AndroidManifest.xml
  The Manifest file described in chapter 2.6

The installed APK files are stored in the /data/app in the mobile file and can be extracted with the adb pull command [44, 45].

2.7 Security

It is not possible to build a 100% secure computer system, security is about trade-offs in which the risk decides the necessity. There are three components in risk evaluations:

- Vulnerability
  Weakness that may result in undesirable consequences
- Threat
  Someone or something with intention to take advantage of a vulnerability
- Consequences
  The result of someone (the threat) using the vulnerability to their advantage

The risk can be thought of as the function of three factors Risk=Consequence (or impact) x Threat x Vulnerability see Figure 3.

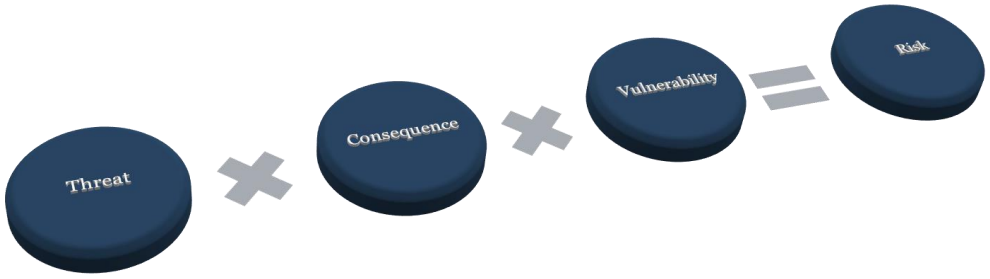

Figure 3 The risk can be described as function

[46]

All applications create vulnerabilities of some sort, but to minimize them there are basically three things to consider confidentiality, integrity, and availability (CIA).

- Confidentiality
  Only the one that supposed to, can get the information and no one else
- Integrity
  It is not possible to manipulate or change the data in any other way than supposed to
- Availability
  Services are available and function as the should

Everything that makes either of these three false, create vulnerabilities that may become a liability if some threat appears [47]. The two area of vulnerabilities concerning android considered in this thesis will be OWSPs definitions of Platform miss usage and insecure storage.

2.8 Platform miss usage

All application installed on a system creates some sort of vulnerabilities. In Android, the applications are by default isolated from each other but most kind of applications need to communicate with the rest of the system to function properly. This often creates security breaches. When misconfigured, the permissions and intent filters of the exported components may give opportunities to malicious applications (see chapter 2.4).

The most common security vulnerabilities today according to OWASP is platform miss usage which often involves exporting components, implementing intent filters and using intents.

OWASP own definition of platform miss usage includes security controls like android intents, TouchID and the keychain (not considered in the thesis). Platform miss usage can often be found by checking against the common guide lines and common conventions. This includes how to export components and how to use intents. Another thing to

consider is semantic errors in which the intention was right but get some part of the implementation wrong [26].

2.8.1 Exported Components

Exported components are declared in the manifest. They are the only way applications can share resources with other applications and they make up much of what in hacker terms is called the “attack surface”. An application can export its components either explicit, or implicitly. To export components implicitly an intent filter may be used. The intent filter exposes a certain set of actions to the system. These can be used by other applications in intent messages(IPC) asking the system for an appropriate action.

If no intent filter is set up exported items must be called explicitly, but that of course means the calling part must know the components name [41].

The exported components and intents represents much of the platform miss usage related vulnerability in applications and will be one of the things to consider in the experiments [26].

2.8.2 Intents

By sending intents to the exported components it’s possible to interact with them. Implicitly called intents get match against intent filters according to the action, data and category.

Intent filter components:

- Action
  The action to be performed
- Data
  Parameters sent along
- Category
  Information about action to execute

It is a security risk to call intents implicitly because there’s no way to be certain of which application answers the call. Because of this the developer’s guideline declare that a service should always be called explicitly [41].

2.8.3 Permissions

Most of the features on the device is protected by permission. To get access the user must accept the uses permission declared in the manifest. This is often done when the applications get installed.

Even though Android defines over 200 permissions they are rather coarse grained and it may be hard for the user to know what they agree on when accepting them [48, 49]. Often free applications have some sort of advertisement service running in the background showing ads, snatching hardware codes and snooping around, sometimes without the consent or knowledge of the user. That is why the applications sometimes asks for permissions like the phone status or get fine location even when it’s obvious it’s not needed for the core function [50, 51].

Table 1 The permissions are divided into three groups

Level

Normal	Usage of data outside sandbox, but with insignificant risk to user privacy or systems integrity (accepted automatically by the system)
Dangerous	Need user's explicit permission when application is installed, risk to privacy and/or may affect system
Signature	Permission granted if signed with the same certificate

The dangerous permissions should be keep at a minimum see Table 2.

Table 2 Dangerous permissions according to Android.developers.com

Type	Permissions
CALENDAR	A. READ_CALENDAR B. WRITE_CALENDAR
CAMERA	A. CAMERA
CONTACTS	A. READ_CONTACTS B. WRITE_CONTACTS C. GET_ACCOUNTS
LOCATION	A. ACCESS_FINE_LOCATION B. ACCESS_COARSE_LOCATION
MICROPHONE	A. RECORD_AUDIO
PHONE	A. READ_PHONE_STATE B. CALL_PHONE C. READ_CALL_LOG D. WRITE_CALL_LOG E. ADD_VOICEMAIL F. USE_SIP G. PROCESS_OUTGOING_CALLS
SENSORS	A. BODY_SENSORS
SMS	A. SEND_SMS B. RECEIVE_SMS C. READ_SMS D. RECEIVE_WAP_PUSH E. RECEIVE_MMS
STORAGE	A. READ_EXTERNAL_STORAGE B. WRITE_EXTERNAL_STORAGE

[52]

2.9 Insecure Data Storage

Insecure storage is the second top 10 OWASP issue. It includes all storage on the device like databases, log files, xml files and the SD card [27]. The android filesystem supplies each application with its own storage place by default isolated from each other. The storage is divided in 2 parts internal and external:

1. Internal
    a. Data/Data/"app-name"
        i. Shared_prefs
            1. Xml files
        ii. Database folder
            1. Db files
        iii. Application resource file

b. Storage/android/app

2. External
   a. SD card/android/app

The external storage can easily be accessed by putting the SD card in another device. Today the some of the newest version of Android encrypts the external storage by default, but it's a convention and good practise to never store any sensitive data on the SD card. Developers often assume data in the applications folder data/data on the device will not be accessible to attackers but this is only true under certain circumstances (not rooted, no backup, not debuggable etc.) so credentials, personal and sensitive data should not be saved on the device at all if it could be avoided. If it can't be avoided the data must be properly encrypted [53]. Insecure storage is one of the OWASP top 10 security issues that will be considered in the experiments [27].

2.9.1 Personal Information PII

Personal identifiable information are information which makes it possible to identify a person as a unique individual see Table 3. If information when compromised can embarrass or otherwise harm the individual it's considered sensitive.

Table 3 Examples of personal information PII

Id	Personal Information PII
1	Social security number
2	ID numbers
3	Combinations of information making it possible to uniquely identify an individual like IP address, name, birth date etc.
5	Telephone number
6	The location and time
7	Visa permit number

Examples of sensitive information might be ethnicity, gender or Information reflecting behaviours and preferences of an individual like web sites visited, searches, religion or sexuality [54, 55, 56].

2.9.2 Hardware Identifiers

Smartphones have just like other hardware unique identification numbers. These numbers can often be used to track or get personal information from the user. There are basically three different numbers that can identify the phone besides the phone number.

- IMEI
  Is a unique hardware identifier of 15 decimal numbers used by smartphones
  The number could be used to track the user, but an even worse issue is that it could be used to blacklisting the phone or clone the number and used it on the black-market.

- SIM
  - IMSI

International Mobile Subscriber Identity is a unique identification number stored in the sim card

- ICCID

A unique number both printed on the sim card and stored inside it. It consists of 19-digit number.

According to google developers guides lines hardware id should not be used to identify the users if it could be avoided. Instead an instance id or an advertisement id should be used [39, 57].

2.10 User Behaviour Vulnerabilities

No system is secure if the user is unaware of the risks involved in certain configurations and behaviours. For example, many people today root to get better control over the phone but as they do so do the hackers and the malicious applications. In a way, these kind of behaviours multiply otherwise small problems by compromising a solid protection of isolation and encryption [40]. Another thing to consider is the settings in the phone, certain settings like USB debugging mode use to debug applications, can with physical access to the phone give unauthorized access [45].

2.10.1 Rooting

One of Linux main core security features is its privilege separation. Linux has two types of accounts normal and root. To install tools and to make changes to the operating system root access is needed. Android is built on top of the Linux kernel and adopt this feature as its own. When buying an android device, the user doesn't have full access to the phone. To gain full access to the phone it must be rooted [58].

People often root their phones when they feel limited by the stock applications or experience poor performance and want to tweak power consumption or look and feel. Rooting the device compromises the security, it breaks the security of the sandbox. Under certain circumstances applications can get root access to system resources [59].

2.10.2 USB debugging Mode

USB debugging mode is a setting in the phones configuration used to access the phone over an USB bridge. It's used by developers to develop their apps in SDK. The combination of rooting and USB debugging mode multiple all risks involved with physical attacks on the phone. Because it makes it possible to get unrestricted access to data on the phone [45].

2.11 Policies and Guidelines

The policies for the developers in Google Play mostly concern type of content and copyright infringement. There are some chapters about how to properly store personal or sensitive information but nothing about intents and exported items java script etc [60]. Developer-Android dot com supplies a check list called core app quality to help assess the core app quality which includes most important of the security tips but those are not enforced by the policies [28].

2.11.1 Google Play Policies

Google have some policies the developers must follow to be allowed to upload to the Google Play store. The policies are divided into various categories:

- Restricted Content
- Intellectual Property, Deception and Spam
  - Impersonation
  - Unauthorised Use of Copyrighted Content
  - Encouraging Infringement of Copyright
  - Trademark Infringement
- Deceptive Behaviour
  - Misleading Claims
  - Unauthorised Use or Imitation of System Functionality
  - Deceptive Device Settings Changes
- Privacy and Security
  - User Data
  - Device and Network
  - Malicious Behaviour
- Monetisation and Ads
  - Payments
  - Subscriptions and Cancellations
  - Ads
  - Authentication
- Store Listing and Promotion
  - App Promotion
  - Metadata
  - Ratings, Reviews and Installs
- Families and COPPA
  - Designed for Families
  - Eligibility
  - Age Groups

Restricted content declares what's allowed in the senses of the content displayed in the applications, like bad languish and sexual content etc. Intellectual Property, Deception and Spam is mostly about copyright but also declare some rules about impersonations in logo name to mislead users. Monetisation and Ads is about what's allowed to sell on the site and Store Listing and Promotion is about in what manner and what kind of advertising allowed. Then there is some privacy and security guide lines about how to handle personal and sensitive information about the user. Lastly Families and COPPA is an attempt to dived the content into age groups to make the site more suitable to children and families [60].

The relevant part of Google Plays guidelines for this thesis is those about privacy and security. It consists of three parts. The first part, user data contains rules for developers about how to handle transfer and store sensitive and personal data. It declares that if an application shares sensitive or personal data it must have a privacy police and handle the data in a secure way both in transit and in storage. If the shared data don't have anything to do with the function of the application it must additionally be transparent about how the data and what data is shared. The other parts of the policy concerning malware and

networking are not included in this thesis because they don't directly concern vulnerabilities in the applications [61, 62].

Google declare on their enforcement page the policy coverage to be defined by DDA (Google Play Developer Distribution Agreement) together with the content rating guidelines and applies to any content the application display or links to.

In DDA Google Play declares what responsibilities and rights developers and Google Play has to remove content from the site that violates the policies. In the 7.2 Google Takedowns:

> "Google does not undertake an obligation to monitor the Products or their content, if Google is notified by you or otherwise becomes aware and determines in its sole discretion that a Product or any portion thereof or your Brand Features;" [61]

They supply a list of violations and continue:

> "Google may remove the Product from Google Play or reclassify the Product at its sole discretion. Google reserves the right to suspend and/or bar any Developer from Google Play at its sole discretion" [61]

2.11.2 App Security Improvement Program

2014 Google Play started the App Security Improvement Program which they now enforce by flagging the issues to the developers giving them a chance to correct the issues before getting vetted. The list of issues can be seen at the web page "App Security Improvement Program"

Besides this Google Play don't dictate how application should be built to be safe and secure. They do not enforce the common guidelines or core qualities defined in android developer.com [61, 60] which means the application could have rather big security holes in any other sense without getting vetted.

2.11.3 Android developers core quality guide lines

The core quality guidelines are a set of rules supposed to be used to test the applications before they are published. These rules are a composition of the most important common android developer guidelines. The rules are divided in five distinct categories:

- Visual design and user interaction
- Functionality
- Compatibility, performance and stability
- Security
- Google Play

[28]

The test sheets used for the vulnerability assessment in method in this thesis will be based on the security part of these guide lines.

3 METHOD

The experiments have three phases starting with a static phase. The statistic phase is meant to open for the dynamic by finding issues that could be exploited. This phase will

focus on finding the attack surface, examine the Android Manifest and if necessary the decompiled and analyse the code see Figure 4.

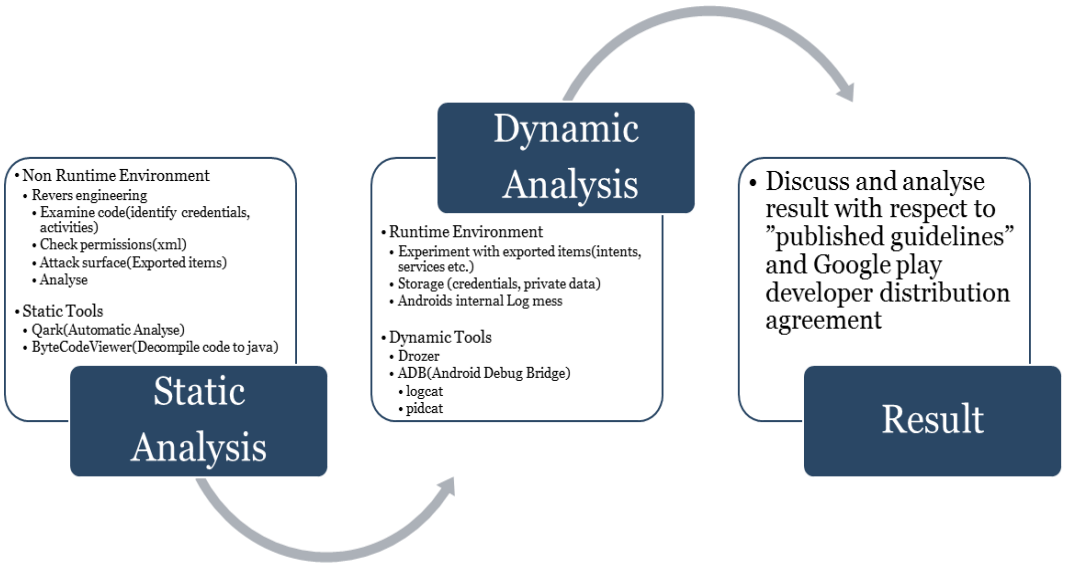

Figure 4 The phases used in the method

The function of the application will be determined and the code will be examined. In the dynamic phase analysis will then be done with the program Drozer and ADB. Drozer lets the user run as an application in runtime and gives opportunity to do most of the things an application would be able to do. Here the static analysis will be used to determine what to look for. The first part of the dynamic analysis the system data log will be examined. To do so the ADB command logcat will be used to examine if any credentials or personal/sensitive information are stored in clear.

a) Where the data is stored
b) If its encrypted

Then Drozer will be used to send intents to the actions in the intent filters to see if it’s possible to exploit or crash the application. Lastly the result will be examined and discussed. The method is not meant as a full coverage test suit but is rather to check in what degree if the applications follow the guide lines.

3.1.1 OWASP

OWSP (Open Web Application Security Project) is a non-profitable organisation started in 2001. It’s an open community dedicated to application security, providing guide lines, tools and forums on the net.

The organisations main project branch applies to web applications but a newly started branch is targeting mobile security risks. Their goal is to help developers to build secure mobile applications by classifying risks and supply forums documents and tools [63].

3.1.2 The core app quality

To get a limited set of tests, relevant according to the common guide lines. The core app quality test sheets from the android developers page will be used. These involves the most

common guidelines and is supposed to be used by the developers before they publish the applications. The test sheet for this method will include the issues in Table 4. The issues excluded was considered too hard to determine or fall outside OWASP top 2 [28]

Table 4 Core app quality tests used in the thesis

Category	Test cases
Security	○ SC-P1 ○ SC-P2 ○ SC-D1 ○ SC-D3 ▪ A ▪ B ○ SC-D4
Functionality	○ FN-P2

3.2 Tools and Platforms

Genymotion is a virtualization platform used to simulate smartphones. By using VirtualBox, Genymotion can run android images of diverse types on top of the OS. Genymotion makes it possible to emulate a rather wide range of smartphones.

The smartphone type used in this thesis is nexus 6 API 23 1440x2560 (marshmallows). This selection was done because android 6 is at the moment (2017) is the most commonly used version of Android.

To download the selected applications, it's necessary to install Google Play. Genymotion smartphone images doesn't come pre-loaded with the Google application packet. To remedy this an installation packet called Gapps (Google apps) was installed on the emulated phone. Gapps is based on an Android modification called CyanogenMod which is compiled in ARM architecture. Because Genymotions ROM is compiled in a x86 architecture it had to be flashed with an ARM architecture before the installation [64, 65].

Most of the tests will be run under command prompt in the win 10 with Drozer and ADB. To be able to use Qark which is not compatible with win10, virtual box will be used to install a virtualized version of Linux Kali.

The main tool used in the static phase is called Qark. Qark (Quick Android Review Kit) is a free to use community based static analysis tool designed for android applications. It analyses the applications to find potential vulnerabilities which then can be validated by dynamic analysis in ADB or Drozer [66]. Qark will be used in the static phase to analyse the applications. The result will be examined and used in the dynamic phase.

In the dynamic phase Drozer will be used. Drozer is an attack frame work used to validate potential vulnerability's. Drozer let the user run as an application in runtime and give opportunity to do most of the things an application would be able to do [67].

Drozer will be used to test the application against a threat (a common term for this is penetration test). In the static analysis, the intents filters and the exported components will be examined to find potential dangerous actions. Intents will then be sent to the exported components to see if it's possible to crash or exploit the application.

The ADB (Android Debug Bridge) is a command line tool swish army knife for android devices. ADB will be used to copy files from the device and to search the system log and the storage for sensitive information. The commands used in this thesis are:

- shell
  Access to the android files from command prompt. Used to search the storage
- logcat
  Prints the system log in the console when application is running, used to check for personal information in system log
- forward
  Port forwarding, connects a port on device to a specific host port
- install
  Used to install APK's from the computer on the device
- pull
  copy files from the device to the host

[45]

To be able to search the database SQLite browser will be used. SQLite browser is a graphical application drag and drop browser for SQLite data bases [68].

To be able to analyse the code it has to be decompiled to Java. The tool used is called ByteCode Viewer. ByteCode Viewer is a graphical application used to revers engineer dex and APK files to java code. Converting dex file in ByteCode Viewer is as simple as drop and drag the files. ByteCode Viewer will be used to analyse the java code in the static phase [69].

Genymotion simulates a rooted phone and may affect the result so this will be taken in consideration when discussing the result.

4 VULNERABILITY ASSESSMENT/PENETRATION TEST

The tests were done according to the test sheet see Table 5. The test sheet is limited to the 2016 OWASPS mobile 2 top issues. The tests are meant to check if the basic core quality guide lines are satisfied in the 10 tested applications concerning top 1 and top 2 issues of OWSPA top10. (The core app quality guide lines are test sheets provided by developer-android dot com with a check list of issues to check before publishing new applications) [70, 28].

4.1 Test sheet

The test sheet consists of 8 static tests and 3 dynamic tests. In the static tests the vulnerabilities will be exposed through finding the attack surface and decompiling and analyse the code. The Android manifest will be examined to determine if the dangerous permissions are necessary to the core function.

Test s4 and s5 which isn't in the core app quality test sheet was picked from the common guide lines because they both makes it much easier to exploit vulnerabilities and in a way multiply otherwise small problems by making data theft much easier see side.

Table 5 The test sheet used in the method

Id	*Static and dynamic analyse*	*Tool/s*	OWASP Top 10	*Core Quality*
S1	1. *Find exported components (attack surface in the context of IPC)* a. *Should be as small as possible* b. *When sharing content with other apps, it should enforce permissions (Actions with data-uri sent along)* 2. *Check for conflicts according to Sc-p2*	*Qark* AndroidManifest.xml	M1	*Sc-p2*
S2	1. *Find dangerous permissions* 2. *What's the usage?* 3. *Put permission grades (low, medium, high) according to necessity* 4. *Conflict according to Fn-p2 if necessity is low (not necessary for core function)* *Perquisite: Understanding functionality of application*	Qark AndroidManifest.xml	M1	FN-P2
S3	1. *Analyse Qark report* 2. *Find implicit pending intents* 3. *Find intent filters with null permission* 4. *Reflect on actions to run in D3* *Possible suggestions in falling order:* 1. *Actions accessing dangerous resources or sensitive data with null permission* 2. *Actions loading webpage (try to load arbitrary page) with null permission* 3. *Actions receiving broadcasts with null permissions* 4. *Run activity to see if it's possible to get passed login screen (if any)* *Perquisite: Attack surface and intent filters, insecure storage*	Qark AndroidManifest.xml Drozer	M1	Sc-d3-a Sc-d3-b
S4	1. *Check for "android: allowBackup" in xml* 2. *Attribute should be set explicitly or else conflict according to general guidelines*	*Qark* *AndroidManifest.xml*	M1	General
S5	1. *Check for "android:Debuggable" in xml* 2. *Attribute should be set to "false" else conflict according to the general guidelines*	*Qark* *AndroidManifest.xml*	M1	General
S6	1. *Decompile APK, analyse code* a. *Obfuscated when billing (General rule)* b. *Hardcoded information Codes, Passwords etc.*	*DataByteCode*	M1, M2	Sc-d1 General
S7	1. *Pull data/data/"appname"/database from device*	*ADB* *Sqlite*	M2	Sc-c1 General

	2. *Check database files for sensitive information (Encryption?)* 3. *Should not contain credentials or personal information without encryption*			
S8	1. *Pull data/data/"appname"/shared_prefs from device* 2. *Pull SD card/data/android/appname from device* 3. *Should not contain credentials or personal information (without encryption)*	*ADB*	M2	Sc-d1
D1	1. *Run logcat with adb* 2. *Check log when:* ***a.*** *logging in* ***b.*** *using search engines* ***c.*** *setting password* 3. *Identify Personal/sensitive Data in log* 4. *If PII personal information or credentials are sent in clear, conflict according to Sc-d4*	*ADB(logcat)*	M2	Sc-d4
D2	• *Check if exported = false explicitly on all hidden components* • *If not conflict according to Sc-p1*	*Drozer*	M1	Sc-p1
D3	*Analysis and selective pen test:* *Reflect on vulnerabilities assessment S1-S8, d1-d2 and Try to exploit at least one of the issues found in the vulnerability assessment by using Drozer to send intents.*	*Drozer*	-	-

4.2 Applications

The applications were randomly picked from Google Plays top list to get the evaluation based on the most downloaded applications. This means that they with high probability have been reviewed properly a thoroughly over some time (making them representable for the application core quality Google Play attempt to keep) see Table 6.

Table 6 The application to be tested randomly picked from the top list at Google Play

Application	Type	Installations	Release	Description
Wish	Web	100 000 000–500 000 000	"4.6.0"	Web store
Eniro	Web	1 000 000–5 000 000	"8.4.3.49.2"	Search engine person's corporations, maps
WWMobile	Web	5 000 000–10 000 000	"5.6.0"	Weight Watchers, web courses and material
Bortskankes	Web	10 000–50 000	"1.45"	Web site for people donating away things

Soundcloud	Web	100 000 000–500 000 000	"2017.04.19-release"	Streaming music, upload your own music
Urbit	Web	10 000–50 000	"1.9.1"	Local Shopping with delivery
Rinkside 3	Web	10 000–50 000	"3.1.5"	Hockey results, chat with other supporters
Pinterst	Web	100 000 000–500 000 000	"6.13.0"	Pins in various categories ideas
Postnord	Web	100 000–500 000	"4.3.1"	Track parcels
Tempelrun	Native	100 000 000–500 000 000	"1.35"	Game

4.3 Result Summery

The summery of the whole pen test can be seen in Table 7. The summery table shows the whole attack surface to the far left and the unprotected part next to it. The "Test fail" column shows how many of the tests had at least one conflict. "Total conflicts" column show all the conflicts found in the application to get a better sense of what's relevant here its divide with the right part from the *D2* test in parentheses because it represents all the hidden components (components of the application with no explicit) which is not explicitly exported (core quality Sc-p1) and it seems it only matters for content providers on devices API<17 which were exported by default. The "crashed test" column shows all the application that crashed in the d3 test and the last column shows the application with vulnerabilities successfully exploited.

Table 7 Summery of pen tests

Application	Attack surface	No Permission	Test fail (test with at least one conflict)	Total Conflicts	Crashed in D3 test	Exploited in D3 test
Wish	12	9	5/10	8+(35)	no	no
Eniro	14	11	5/10	7+(52)	yes	no
WWMobile	16	11	2/10	4+(93)	no	no
Bortskankes	1	0	2/10	2+(0)	no	no
Soundcloud	26	22	3/10	11+(55)	no	yes
Urbit	11	10	5/10	11+(36)	no	no
Rinkside3	5	4	3/10	4+(2)	yes	no
Pinterest	16	13	3/10	3+(32)	no	no
Postnord	15	13	3/10	5+(14)	no	no
Templerun	4	4	4/10	6+(18)	yes	no

4.4 Result Analysis

The result shows that all the tested applications break against the rules in the core app quality sheet in several ways. Some of the rules/tests in the list may be overkill and others a bit silly but it still indicates inferior quality from a security point of view. Out of 10 tested applications the D3 tests (simple test sending intents from Drozer to the applications)

crashed three applications and found a rather serious flaw possible to exploit in one. Taking into consideration that the test was very simple with just a few intents sent make one wonder what would have been exposed in a full coverage test or intent fuzzing.

The exploit in SoundCloud was possible because a combination of insecure storage and improper platform usage. The application exposes a system resource (the microphone) to the entire system without enforcing permission and then by default saves the recording to the SD card.

It's possible to start the recording with an intent without signing in to the user account. The service handling the recording then runs in the background unnoticeable. If the SD card isn't encrypted. Anyone with physical access to the phone could extract the recordings by removing the card and putting it in another device. Beside recording audio, the application can upload its recordings to the web site and put them in an either public or private library. This too seems to be possible to do by sending intents using the two actions SEND and SHARE which would make the matter even worse (this was not verified in the tests).

Another flaw relating to the external storage was found in the Wish web discount store. The Wish application caches all the pics it comes across when the user search the shop on the SD card in clear. According to PII (see chapter 2.9.1) the pictures alone isn't enough to be classified as sensitive information but together with identifiable information they might be.

Besides that, many minor security flaws like personal information in clear stored at the wrong place, asking for unnecessary dangerous permissions (mostly camera) and personal information in the system log was found in several applications see chapter 9.1.

4.4.1 Google Plays policies

The fact that Wish (100 000 000–500 000 000 downloads) puts the all the pictures from the web shop searches on the SD card in clear seems to be fine with Google Plays policy. Even if their guidelines declare sensitive and personal data should be encrypted if its stored on the SD card. The pictures can only be classified as sensitive personal information if the individual is uniquely identified (see chapter 2.9.1) and that may or may not be the case depending on the situation.

The fact that applications crash when sending intents with arbitrary input to them isn't an issue according to Google Plays policy. Even though it makes the applications vulnerable to attacks in several ways.

According to Google Plays policies (in user data) it's not allowed to put sound recordings from the microphone on the SD card in clear as it is an insecure storage place [71]. But still SoundCloud an application which have 100 000 000–500 000 000 downloads is allowed to do so without getting vetted.

5 DISCUSSION/CONCLUSION

According to the tests it seems like Google Play store has many applications that breaks against the core app quality guidelines. And that it sometimes leads to vulnerabilities possible to exploit.

But to make a good evaluation of what the risk on Google Play might be all three of the aspects in risk evaluation must be considered (see chapter 2.7). In this thesis only the vulnerabilities and consequence were accounted for. The level of threat in this case greatly depends on in what degree Google Play find certain kinds of threats related to the problems in the applications tested in the thesis. But it's unlikely that Google Play Protect can find applications exploiting normal usage of intents and actions even if they are used in malicious way. So why then doesn't Google Play vet applications when they break against their policies?

In the Google Play Developer Distribution Agreement paragraph 7.2, Google Takedowns, Google declare they don't undertake any obligation to monitor the products or their content. So even if an application according to Google Plays own policies should be vetted. It would have to be on the customers initiative and then only if Google Play decide to act upon it which is their choice or like they declare in DDA "Google may at its discretion disable the Product" [61].

Accordingly, the SoundCloud application which clearly breaks against Google Plays policy might never be vetted. Because the customers who apparently asked for the feature to save the recordings on the SD card themselves [72]. Are unlikely to know anything about the vulnerability making it possible for another application to start and stop the microphone (without their consent). And are probably unwilling to complain over a feature they asked for themselves.

In this context, it's important that the developers take their responsibility because Android is an open permissive operating system that trusts the users and the developers to play fair and follow the rules. But only Google Play can enforce a real overall difference because it is through their policies and guidelines the standard is set.

6 PROBLEMS/ISSUES

- The tests might have been flawed in many ways.
    - The application WWMobile was a bit hard to test due to being a paid service. It wasn't obvious when it was randomly selected and it probably would have been better to change application but it wasn't exposed until several of the tests already was done (Qark report etc.) which implies the test was done in wrong order (S1 and S2) but the Qark report was used to get the xml.
    - The experiments were executed on an emulated android phone and may be somewhat misleading, in a real phone provided it's not rooted and not in debug mode it might been harder to exploit the flaws found.
    - The test sheets for the method uses a combination of the Android developers web site called core app quality delimited by the first two of OWSASP mobile top 10 securities issues. This limitation may affect the result depending on how the security issues in the applications are distributed (more or less in certain areas).
    - Too few experiments to draw accurate conclusions.
    - In the method, all core quality conflicts count as if they are valued same, when the truth may be that some of them counts less or more when considering the vulnerability. Because of that the result of the method can't be used as a true vulnerabilities assessment and is rather used as an

indication of how much the common guidelines is considered and in what degree the developers test the applications against the issues in the core app quality test sheet.
    - Many of the tests must be interpreted by someone and can have different interpretations
        - S2 test is based on finding what might or not might be a core function of the application and could many times be interpreted differently. The function has been checked and the necessity of the permission has been evaluated after best effort.
        - Many test checks if for sensitive information which sometimes may have different interpretations.
- The Qark report contained a lot of useful information about vulnerabilities but much of it wasn't tested against a threat due to the limitation of this thesis.
    - One aspect that is in the limitation of the thesis but wasn't tested was the local pending intents, many applications use them extensively but they are according to android developers.com dangerous and should always call explicitly when the class is known (SC-d3 a). No attacks were conducted against this vulnerability which was present in almost all the applications.

7 FUTURE WORK

It would be very interesting to examine what the correlation between issues like crashes and the number of exported components is (maybe have a look on the ratio between exported components with no permission and all the exported components) by running intent fuzzing on the applications.

8 REFERENCES

[1] Statista, "Number of smartphone users worldwide from 2014 to 2020 (in billions)," [Online]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/. [Accessed 02 04 2017].

[2] T. Martin, "The evolution of the smartphone," 28 07 2014. [Online]. Available: http://pocketnow.com/2014/07/28/the-evolution-of-the-smartphone. [Accessed 03 06 2017].

[3] T. Gillenwater, "Evolution of the Smartphone," 14 02 2017. [Online]. Available: http://www.microwavejournal.com/articles/27780-evolution-of-the-smartphone. [Accessed 03 06 2017].

[4] J. Hildenbrand, "Android 7.0: Security benefits that truly matter," 2016. [Online]. Available: https://www.androidcentral.com/how-android-n-addresses-security. [Accessed 24 04 2017].

[5] C. Hoffman, “The Case Against Root: Why Android Devices Don’t Come Rooted,” 31 12 2012. [Online]. Available: https://www.howtogeek.com/132115/the-case-against-root-why-android-devices-dont-come-rooted/. [Accessed 26 05 2017].

[6] Google, “Help protect against harmful apps,” [Online]. Available: https://support.google.com/accounts/answer/2812853?hl=en. [Accessed 01 06 2017].

[7] I. Barker, “69 percent of users would bypass security controls to win a big deal,” 2015. [Online]. Available: https://betanews.com/2015/11/20/69-percent-of-users-would-bypass-security-controls-to-win-a-big-deal. [Accessed 23 05 2017].

[8] A. Strategy, “Strategy Analytics Press Releases,” 02 11 2016. [Online]. Available: https://www.strategyanalytics.com/strategy-analytics/news/strategy-analytics-press-releases/strategy-analytics-press-release/2016/11/02/strategy-analytics-android-captures-record-88-percent-share-of-global-smartphone-shipments-in-q3-2016#.WRlX9cYlGUk. [Accessed 02 04 2017].

[9] R. Triggs, “The five reasons why Android is killing Apple,” 28 06 2013. [Online]. Available: http://www.androidauthority.com/five-reasons-android-killing-apple-234364. [Accessed 02 04 2017].

[10] S. Hill, “Android app security basics: Easy ways to keep your phone safe,” 01 09 2014. [Online]. Available: https://www.digitaltrends.com/android/android-app-security-basics/. [Accessed 25 05 2017].

[11] Google, “Google Account help security,” [Online]. Available: https://support.google.com/accounts/answer/2812853?hl=en. [Accessed 29 05 2017].

[12] G-data, “8,400 new Android malware samples every day,” 29 04 2017. [Online]. Available: https://blog.gdatasoftware.com/2017/04/29712-8-400-new-android-malware-samples-every-day. [Accessed 25 05 2017].

[13] G. Kelly, “Report: 97% Of Mobile Malware Is On Android. This Is The Easy Way You Stay Safe,” 24 03 2014. [Online]. Available: https://www.forbes.com/sites/gordonkelly/2014/03/24/report-97-of-mobile-malware-is-on-android-this-is-the-easy-way-you-stay-safe/#14c4ce702d4f. [Accessed 26 05 2017].

[14] Statista, “Number of available applications in the Google Play Store from December 2009 to March 2017,” 2017. [Online]. Available: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store. [Accessed 01 04 2017].

[15] M. R. B. C. D. H. C. Mahmudur Rahman, “FairPlay: Fraud and Malware Detection in Google Play,” Florida Int’l Univ, 2015.

[16] M. Y. B. X. Z. Y. G. G. P. N. S. W. B. Z. Yuan Zhang, “Vetting Undesirable Behaviours in Android Apps with Permission Use Analysis,” Fudan University, China, Fudan, 2014.

[17] IDC, “Worldwide Smartphone Growth Forecast to Slow to 3.1% in 2016 as Focus Shifts to Device Lifecycles, According to IDC,” 01 06 2016. [Online]. Available: http://www.idc.com/getdoc.jsp?containerId=prUS41425416. [Accessed 02 04 2017].

[18] G. Duncan, “Is Google helpless to stop the scourge of Android malware?,” 03 01 2013. [Online]. Available: https://www.digitaltrends.com/android/who-can-fight-android-malware-not-google/. [Accessed 24 05 2017].

[19] Google, “How we keep harmful apps out of Google Play and keep your Android device safe,” Google inc, 2016.

[20] Google, “Android Security 2016 Year in Review,” Google, 2017.

[21] J. Leyden, “Android device manager app vuln leaves millions at risk of pwnage,” 19 02 2016. [Online]. Available: https://www.theregister.co.uk/2016/02/19/airdroid_vuln. [Accessed 16 04 2017].

[22] L. Constantin, “Google pushed developers to fix security flaws in 275,000 Android apps,” 20 01 2017. [Online]. Available: http://www.pcworld.com/article/3159972/security/google-pushed-developers-to-fix-security-flaws-in-275000-android-apps.html. [Accessed 28 04 2017].

[23] D. Android, “App Security Improvement Program,” [Online]. Available: https://developer.android.com/google/play/asi.html#campaigns. [Accessed 10 04 2017].

[24] C. Smith, “obscure-app-flaw-creates-backdoors-millions-smartphones,” 09 05 2017. [Online]. Available: https://www.wired.com/2017/04/obscure-app-flaw-creates-backdoors-millions-smartphones/. [Accessed 03 06 2017].

[25] C. Smith, “Google isn’t fixing a serious Android security flaw for months,” 09 05 2017. [Online]. Available: http://bgr.com/2017/05/09/android-permissions-security-flaw-android-o/. [Accessed 29 04 2017].

[26] OWASP, “Mobile Top 10 2016-M1-Improper Platform Usage,” 06 03 2017. [Online]. Available: https://www.owasp.org/index.php/Mobile_Top_10_2016-M1-Improper_Platform_Usage. [Accessed 03 05 2017].

[27] OWASP, “Mobile Top 10 2016-M2-Insecure Data Storage,” [Online]. Available: https://www.owasp.org/index.php/Mobile_Top_10_2016-M2-Insecure_Data_Storage. [Accessed 03 06 2017].

[28] D. Android, “Core app quality,” [Online]. Available: https://developer.android.com/develop/quality-guidelines/core-app-quality.html. [Accessed 15 05 2017].

[29] OpenHandsetAlliance, “Android,” [Online]. Available: https://www.openhandsetalliance.com/android_overview.html. [Accessed 03 06 2017].

[30] Statista, “global-market-share-held-by-smartphone-operating-systems,” 2017. [Online]. Available: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/. [Accessed 27 05 2017].

[31] C. Arthur, “The history of smartphones: timeline,” 24 01 2012. [Online]. Available: https://www.theguardian.com/technology/2012/jan/24/smartphones-timeline. [Accessed 26 05 2017].

[32] C. Hoffman, “Android Is “Open” and iOS Is “Closed” — But What Does That Mean to You?,” 24 05 2015. [Online]. Available: https://www.howtogeek.com/217593/android-is-open-and-ios-is-closed-but-what-does-that-mean-to-you/. [Accessed 28 05 2017].

[33] D. Android, Artist, *Android software stack.* [Art]. Google.

[34] A. D. Anmol Misra, Android Security,Android Architecture, CRC Press, 2016.

[35] J. Six, “Application Security for the Android Platform, The Linux Security Model,” John Wiley & Sons, 2011.

[36] J. Six, “Application Security for the Android Platform, The Resulting Android Security Model,” O'Reilly Media, 2011.

[37] A. Misra and A. Dubey, “Android Security,Android Application Architectur,” Auerbach Publications, 2015.

[38] D. Android, “App Manifest,” [Online]. Available: https://developer.android.com/guide/topics/manifest/manifest-intro.html. [Accessed 23 05 2017].

[39] D. Android, “Security Tips,” [Online]. Available: Security Tips https://developer.android.com/training/articles/security-tips.html. [Accessed 23 05 2017].

[40] M. A. I. Srinivasa Rao Kotipalli, “Hacking Android,Introduction to Android apps,” Packt Publishing, 2016.

[41] D. Android, “Intents and Intent Filters,” [Online]. Available: https://developer.android.com/guide/components/intents-filters.html. [Accessed 28 04 2017].

[42] D. Android, “AndroidManifest.xml,” [Online]. Available: https://developer.android.com/samples/BasicContactables/AndroidManifest.html. [Accessed 26 05 2017].

[43] D. Android, “Requesting Permissions at Run Time,” [Online]. Available: https://developer.android.com/training/permissions/requesting.html. [Accessed 02 06 2017].

[44] S. R. Kotipalli and M. A. Imran, “Hacking Android, Android app structure,” Packt Publishing, 2016.

[45] D. Android, “Android Debug Bridge,” [Online]. Available: https://developer.android.com/studio/command-line/adb.html. [Accessed 19 05 2017].

[46] J. Six, “Application Security for the Android Platform, Security: Risk = Vulnerability + Threat + Consequences,” O'Reilly Media, 2011.

[47] J. Six, “Application Security for the Android Platform, Evolution of Information Security: Why Applications Matter the Most,” O'Reilly Media, 2011.

[48] D. Android, “Manifest.permission,” [Online]. Available: https://developer.android.com/reference/android/Manifest.permission.html. [Accessed 03 06 2017].

[49] N. B. P. R. V. G. L. I. Liu Yang, “Enhancing Users’ Comprehension of Android,” The State University of New Jersey, New Jersey, 2012.

[50] A. A. W. D. Xiao Zhang, “AFrame: Isolating Advertisements from Mobile,” Syracuse University, Syracuse, 2013.

[51] D. Goodin, “More Android phones than ever are covertly listening for inaudible sounds in ads,” 05 05 2017. [Online]. Available: https://arstechnica.com/security/2017/05/theres-a-spike-in-android-apps-that-covertly-listen-for-inaudible-sounds-in-ads/. [Accessed 02 06 2017].

[52] D. Android, “Requesting Permissions,” [Online]. Available: https://developer.android.com/guide/topics/ permissions/requesting.html. [Accessed 19 05 2017].

[53] J. Six, “Application Security for the Android Platform, The Threats and Vulnerabilities Against Stored Data,” O'Reilly Media, 2011.

[54] Investopedia, “Personally Identifiable Information (PII),” [Online]. Available: http://www.investopedia.com/terms/p/personally-identifiable-information-pii.asp. [Accessed 29 05 2017].

[55] I. C. Office, “Key definitions of the Data Protection Act,” [Online]. Available: https://ico.org.uk/for-organisations/guide-to-data-protection/key-definitions/. [Accessed 30 05 2017].

[56] T. U. o. Manchester, “Processing sensitive personal data,” [Online]. Available: http://www.dataprotection.manchester.ac.uk/whatisdataprotection/sensitivepersonaldata/. [Accessed 30 05 2017].

[57] A. S. P. G. Fabio Casadei, “Forensics and SIM cards: an Overview,” *International Journal of Digital Evidence,* vol. 5, no. 1, 2006.

[58] S. R. Kotipalli and M. A. Imran, “Hacking Android, What is rooting?,” Packt Publishing, 2016.

[59] T. Phelps, “To Root or Not to Root Android,” 13 03 2017. [Online]. Available: https://www.lifewire.com/root-or-not-root-android-1616838. [Accessed 24 05 2017].

[60] Google, “Policycenter för utvecklare,” [Online]. Available: https://play.google.com/about/developer-content-policy-print. [Accessed 05 05 2017].

[61] Google, “Google Play Developer Distribution Agreement,” 17 05 2017. [Online]. Available: https://play.google.com/about/developer-distribution-agreement.html. [Accessed 02 06 2017].

[62] Google, “Enforcement,” [Online]. Available: https://play.google.com/intl/en-GB/about/enforcement/index.html. [Accessed 05 05 2017].

[63] OWASP, “OWASP Mobile Security Project,” 27 04 2017. [Online]. Available: https://www.owasp.org/index.php/OWASP_Mobile_Security_Project. [Accessed 28 04 2017].

[64] Techbae, “How To Download & Install ARM Translation v1.1.zip for Genymotion,” 08 10 2015. [Online]. Available: http://www.techbae.com/download-install-arm-translation-v1-1-zip-genymotion/. [Accessed 04 06 2017].

[65] Xda-developers, “How To Download and Install Google Apps,” [Online]. Available: https://www.xda-developers.com/download-google-apps-gapps/. [Accessed 03 06 2017].

[66] I. Infosec, “QARK – A tool for automated Android App Assessment,” 05 10 2015. [Online]. Available: http://resources.infosecinstitute.com/qark-a-tool-for-automated-android-app- assessments. [Accessed 22 04 2017].

[67] Mwrinfosecurity, “Drozer,” [Online]. Available: https://labs.mwrinfosecurity.com/tools/drozer/. [Accessed 24 05 2017].

[68] Sqlitebrowser, “DB browser for sqlite,” [Online]. Available: http://sqlitebrowser.org/. [Accessed 25 05 2017].

[69] Konloch, “Bytecode Viewer,” [Online]. Available: http://bytecodeviewer.com/. [Accessed 21 05 2017].

[70] OWASP, “Mobile Top 10 2016-Top 10,” [Online]. Available: https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10. [Accessed 29 05 2017].

[71] Google, "Privacy and Security," [Online]. Available: https://play.google.com/intl/en-US/about/privacy-security/index.html. [Accessed 23 05 2017].

[72] Soundcloud, "Soundcloud Go Storage Location," 2016. [Online]. Available: https://soundcloudcommunity.com/soundcloud-on-your-phone-tablet-230063/soundcloud-go-storage-location-6966266. [Accessed 28 05 2017].

9 APPENDIX

9.1 Tests

9.1.1 Wish

Table 8 Wish

<table>
<tr><th>Test</th><th>Appendix</th><th>Test outcome/Comments</th><th>Guideline Conflict</th></tr>
<tr><td>S1</td><td></td><td>
<table>
<tr><td rowspan="2">Type of component</td><td colspan="2">Exported</td></tr>
<tr><td>No Permission</td><td>Permission</td></tr>
<tr><td>Activity</td><td>5</td><td>0</td></tr>
<tr><td>Content Provider</td><td>0</td><td>0</td></tr>
<tr><td>Service</td><td>1</td><td>1</td></tr>
<tr><td>Receiver</td><td>3</td><td>2</td></tr>
<tr><td>Total</td><td>9</td><td>3</td></tr>
</table>
Attack surface: 12
Null permission: 9
</td><td>0</td></tr>
<tr><td>S2</td><td></td><td>
<table>
<tr><td>Hard ware Uses-Permission</td><td>Usage</td><td>Necessity</td></tr>
<tr><td>CAMERA</td><td>Face pic in</td><td>Low</td></tr>
<tr><td>WRITE_EXTERNAL_STORAGE</td><td>cache</td><td>medium</td></tr>
<tr><td>READ_EXTERNAL_STORAGE</td><td>cache</td><td>medium</td></tr>
<tr><td>RECEIVE_SMS</td><td>buying</td><td>high</td></tr>
<tr><td>GET_ACCOUNTS</td><td>login</td><td>medium</td></tr>
<tr><td>READ_CONTACTS</td><td>Facebook</td><td>low</td></tr>
</table>
At least two of the permissions seems to be unnecessary. The only use of the camera seems to be to take profile photos which could be done outside the application.
</td><td>2</td></tr>
<tr><td>S3</td><td>Error! Reference source not found.</td><td>
<table>
<tr><td>Issue</td><td></td></tr>
<tr><td>Pending implicit local intents</td><td>4</td></tr>
<tr><td>Intent filters no permission</td><td>11</td></tr>
<tr><td>Total</td><td>15</td></tr>
</table>
Test in D3:
• action to run
○ com.contextlogic.wish.DEVICE_ID
May be a hardware identifier
</td><td>4</td></tr>
</table>

<table>
<tr><td></td><td></td><td>android.intent.action.VIEW
Sometimes possible to load an arbitrary page with malicious code
com.android.vending.INSTALL_REFERRER</td><td></td></tr>
<tr><td>S4</td><td></td><td>True</td><td>0</td></tr>
<tr><td>S5</td><td></td><td>False</td><td>0</td></tr>
<tr><td>S6</td><td></td><td>Billing not obfuscated
Sends the uuid by broadcast receiver without permission, seems to be the advertisement id,,,,, ok!!!!
riskifiedbecon???? it tries to send imei and other personal information (which isn’t possible in this case due to no phone status permission) seems to be googles own class???</td><td>1</td></tr>
<tr><td>S7</td><td></td><td>Database seems to be empty?
Don’t know what’s stored here?</td><td>0</td></tr>
<tr><td>S8</td><td>Figure 5</td><td>Shared_prefs:
Found user name and device UUID in clear (probably not a problem)
Braintree encrypted credit card info (ok known billing solutions company)
SD card:
Cached pics of old searches, privacy concern see appendix fig</td><td>1</td></tr>
<tr><td>D1</td><td></td><td>Checked login, and searches
No issues found</td><td>0</td></tr>
<tr><td>D2</td><td></td><td><table><tr><td rowspan="2">Type of component</td><td colspan="2">Hidden</td></tr><tr><td>Explicitly</td><td>Implicitly</td></tr><tr><td>Activity</td><td>2</td><td>35</td></tr><tr><td>Content Provider</td><td>2</td><td>0</td></tr><tr><td>Service</td><td>2</td><td>0</td></tr><tr><td>Receiver</td><td>1</td><td>0</td></tr><tr><td>Total</td><td>7</td><td>35</td></tr></table>35 hidden with default implicit export = false</td><td>35</td></tr>
<tr><td>D3</td><td>Table 18</td><td><table><tr><td>Id</td><td>Possible exploit</td><td>Outcome</td></tr><tr><td>SE1</td><td>Is it possible to get the device id
What kind of id is it?</td><td>No, the id is sent somewhere, can’t see it in log, advertisement?
From analyse in bytecodeviewer it seems to be an advertisement id(ok)</td></tr><tr><td>SE2</td><td>Is it possible to open an</td><td>No, open same page even when</td></tr></table></td><td>0</td></tr>
</table>

			arbitrary webpage from web view	html sent along as uri		
		SE3	1. Is it possible to open face book from intent?	1. No, open default page even when html sent along as uri		

9.1.2 Eniro

Table 9 Eniro

Test	Appendix	Comments	*Guideline Conflict*
S1	Figure 10	Type of component / Exported (No Permission / Permission) Activity: 2 / 0 Content Provider: (1) see fig* / 0 Service: 1 / 1 Receiver: 8 / 2 Total: 11 / 3 The application got 11 exported components with null permission and 1 content provider(null) sharing data which result in a conflict according to guidelines. Attack surface: 14 Null permission: 11	(1)
S2		Hard ware Uses-Permission / Usage / Necessity ACCESS_FINE_LOCATION / site-adapted information and marketing / medium CALL_PHONE / Phone from within eniro app / medium READ_CALL_LOG / See call log within eniro / medium READ_CONTACTS / Caller-id / medium READ_EXTERNAL_STORAGE / cache / medium WRITE_EXTERNAL_STORAGE / cache / medium READ_PHONE_STATE / Look up phone number ect. / medium RECEIVE_SMS / ? / ?	1

<table>
<tr><td></td><td></td><td><table><tr><td>SYSTEM_ALERT_WINDOW</td><td>?</td><td>should not use at all!!!</td></tr><tr><td>WRITE_EXTERNAL_STORAGE</td><td></td><td>?</td></tr></table></td><td></td></tr>
<tr><td>S3</td><td>Figure 9</td><td><table><tr><td>Issue</td><td>Null Permission</td></tr><tr><td>Pending implicit local intents</td><td>3</td></tr><tr><td>Intent no permission</td><td>11</td></tr><tr><td>Total</td><td>14</td></tr></table>
Test in D3:
action to run
○ android.intent.action.VIEW</td><td>3</td></tr>
<tr><td>S4</td><td></td><td>True</td><td>0</td></tr>
<tr><td>S5</td><td></td><td>no</td><td>0</td></tr>
<tr><td>S6</td><td>Figure 8</td><td>• Is obfuscated
• Hardcoded password see fig
• Hardcoded username see fig
(to webpage emmadey.cloudant.com)</td><td>2</td></tr>
<tr><td>S7</td><td></td><td>• Database dir:
Plain text in history (backup enabled makes it accessible from adb)
• Files dir:
Eniro-db,eniro-cache and Jason files encrypted need pass to open(ok)</td><td>0</td></tr>
<tr><td>S8</td><td></td><td>• Shared_prefs:
ok
• Sd card:
Cache in some sort of Picasso(fine)</td><td>0</td></tr>
<tr><td>D1</td><td></td><td>Ok</td><td></td></tr>
<tr><td>D2</td><td></td><td><table><tr><td rowspan="2">Type of component</td><td colspan="2">Hidden</td></tr><tr><td>Explicitly</td><td>Implicitly</td></tr><tr><td>Activity</td><td>1</td><td>18</td></tr><tr><td>Content Provider</td><td>3</td><td>0</td></tr><tr><td>Service</td><td>1</td><td>33</td></tr><tr><td>Receiver</td><td>2</td><td>1</td></tr><tr><td>Total</td><td>7</td><td>52</td></tr></table></td><td>52</td></tr>
<tr><td>D3</td><td>Table 19</td><td><table><tr><td>Id</td><td>Possible exploit</td><td>Outcome</td></tr><tr><td>SE1</td><td>1. Is it possible to open an arbitrary webpage from web view in com.eniro com.eniro.core. android.map.MapActivity</td><td>1. No The but the program crash! (The program should not crash, all exceptions</td></tr></table></td><td></td></tr>
</table>

				should be handled!)	

9.1.3 WWMobile

Table 10 WWMobile

Test	Appendix	Comments	*Guideline Conflict*
S1		Type of component: Exported (No Permission / Permission) Activity: 8 / 1 Content Provider: 0 / 0 Service: 2 / 1 Receiver: 1 / 3 Total: 11 / 5 The application has no content providers, and the exported items don't seem to share content with each other so according to the guide lines this is ok. One could argue that the attack surface should be as small as possible, but it's hard to tell if there's an issue in this case from just a shallow analyse of the application. Attack surface: 16 Null permission: 11	0
S2		Hard ware Uses-Permission / Usage / Necessity ACCESS_COARSE_LOCATION: Adapt to user / medium CALL_PHONE: ? / ? CAMERA: ? / ? GET_ACCOUNTS: ? / ? READ_CONTACTS: ? / ? READ_EXTERNAL_STORAGE: ? / ? WRITE_EXTERNAL_STORAGE: ? / ? READ_PHONE_STATE: ? / ? The application costs money, can't sign in which wasn't clear from the start, because of that it's hard to tell if the permissions are core functions or not.	-
S3	Figure 11	Issue / Conflict Pending implicit local intents: 4 Intent no permission: 11 Total: 15 **Test in D3:**	4

<table>
<tr><td></td><td></td><td>action to run
• android.intent.action.VIEW
• com.weightwatchers.food.action.SEARCH</td><td></td></tr>
<tr><td>S4</td><td></td><td>True</td><td>0</td></tr>
<tr><td>S5</td><td></td><td>no</td><td>0</td></tr>
<tr><td>S6</td><td></td><td>• Not obfuscated didn’t find anything can’t find provider check again if time
• Get IMEI to create something called persistent UUID(ok not a hardware id)</td><td>0</td></tr>
<tr><td>S7</td><td></td><td>Nothing stored yet can’t login cost money</td><td>0</td></tr>
<tr><td>S8</td><td></td><td>• Shared_prefs: Credentialstore.xml credentials encrypted ok</td><td>0</td></tr>
<tr><td>D1</td><td></td><td>Nothing personal in log</td><td>0</td></tr>
<tr><td>D2</td><td></td><td><table>
<tr><td rowspan="2">Type of component</td><td colspan="2">Hidden</td></tr>
<tr><td>Explicitly</td><td>Implicitly</td></tr>
<tr><td>Activity</td><td>4</td><td>88</td></tr>
<tr><td>Content Provider</td><td>0</td><td>0</td></tr>
<tr><td>Service</td><td>4</td><td>2</td></tr>
<tr><td>Receiver</td><td>0</td><td>3</td></tr>
<tr><td>Total</td><td>8</td><td>93</td></tr>
</table></td><td>93</td></tr>
<tr><td>D3</td><td>Figure 12
Table 20</td><td><table>
<tr><td>Id</td><td>Possible exploit</td><td>Outcome</td></tr>
<tr><td>SE1</td><td>1. Is it possible to open an arbitrary webpage from web view see test 1,2 3, in table in appendix</td><td>1. No
Get message not able to find this member
2. No
Just flash the screen and gets back to sign in
3. No
Get message doesn’t seem to find any items</td></tr>
</table></td><td></td></tr>
</table>

9.1.4 Bortskankes

Table 11 Bortskankes

<table>
<tr><th>Test</th><th>Appendix</th><th>Comments</th><th>Guideline Conflict</th></tr>
<tr><td>S1</td><td></td><td><table>
<tr><td rowspan="2">Type of component</td><td colspan="2">Exported</td></tr>
<tr><td>No Permission</td><td>Permission</td></tr>
<tr><td>Activity</td><td>1</td><td>0</td></tr>
<tr><td>Content Provider</td><td>0</td><td>0</td></tr>
<tr><td>Service</td><td>0</td><td>0</td></tr>
<tr><td>Receiver</td><td>0</td><td>0</td></tr>
</table></td><td>0</td></tr>
</table>

		<table><tr><td>Total</td><td>1</td><td>0</td></tr></table> The application has just one activity which must be exported to interact, ideal but hard to achieve in a more complex application. Attack surface: 1	
S2		<table><tr><td>Hard ware Uses-Permission</td><td>Usage</td><td>Necessity</td></tr><tr><td>ACCESS_FINE_LOCATION</td><td>?</td><td>low</td></tr><tr><td>ACCESS_COARSE_LOCATION</td><td>adapts</td><td>medium</td></tr><tr><td>WRITE_EXTERNAL_STORAGE</td><td>cache</td><td>medium</td></tr><tr><td>READ_EXTERNAL_STORAGE</td><td>cache</td><td>medium</td></tr></table> Can't find any usage of fine location but it's not necessary for core function.	1
S3		No pending intents No intent filters	0
S4		Default	1
S5		no	0
S6		• Not obfuscated • No login so no password, can't find anything hardcoded Only one component nothing much to say	0
S7		• C:\Apps\data2\app_webview Contains some interesting databases with credit card saved in web_data, is encrypted don't know with what kind of encryption	0
S8		• data/data ○ Cache saved (some sort of Picasso=not plain ok) ○ Nothing much in shared_prefs ○ cookies	0
D1		Can't find anything here	0
D2		No hidden components exported by default	0
D3		Only one component, not much to run	-

9.1.5 Soundcloud

Table 12 SoundCloud

Test	Appendix	Result	Guideline Conflict
S1		<table><tr><td rowspan="2">Type of component</td><td colspan="2">Exported</td></tr><tr><td>No Permission</td><td>Permission</td></tr><tr><td>Activity</td><td>12</td><td>0</td></tr><tr><td>Content Provider</td><td>0</td><td>0</td></tr><tr><td>Service</td><td>2</td><td>1</td></tr><tr><td>Receiver</td><td>8</td><td>3</td></tr><tr><td>Total</td><td>22</td><td>4</td></tr></table>	0

<table>
<tr><td></td><td></td><td>The application has no content providers, and the exported items don’t seem to share content with each other so according to the guide lines this is ok. One could argue that the attack surface should be as small as possible.

Attack surface: 26
Null permission: 22

Very large attack surface with most of it exposed without permissions.</td><td></td></tr>
<tr><td>S2</td><td></td><td><table>
<tr><td>Hard ware Uses-Permission</td><td>Usage</td><td>Necessity</td></tr>
<tr><td>RECORD_AUDIO</td><td>Upload sound</td><td>high</td></tr>
<tr><td>GET_ACCOUNTS</td><td>login</td><td>medium</td></tr>
<tr><td>WRITE_EXTERNAL_STORAGE</td><td>Store audio</td><td>medium</td></tr>
<tr><td>READ_EXTERNAL_STORAGE</td><td>Store audio</td><td>medium</td></tr>
</table>
Record audio seem to be central to the functions but it might be better to let the user record outside the application and access the files from within the application. Using the SD card seem logical, the audio files probably take some space. Get accounts to log in convenient but unnecessary.</td><td>0</td></tr>
<tr><td>S3</td><td></td><td><table>
<tr><td>Issue</td><td></td></tr>
<tr><td>Pending implicit local intents</td><td>10</td></tr>
<tr><td>Intent no permission</td><td>31</td></tr>
<tr><td>Total</td><td>10</td></tr>
</table>
Test in D3:
action to run
• com.soundcloud.android.creators.record.RecordActivity
○ com.soundcloud.android.action.RECORD_START
• com.soundcloud.android.creators.
record.UploadActivity
○ android.intent.action.SEND</td><td>10</td></tr>
<tr><td>S4</td><td></td><td>False</td><td>0</td></tr>
<tr><td>S5</td><td></td><td>False</td><td>0</td></tr>
<tr><td>S6</td><td></td><td>• Billing obfuscated
• No hardcoded credentials
• Encrypted</td><td>0</td></tr>
<tr><td>S7</td><td></td><td>• Adlog
Location time stored?
https://www.appboy.com/documentation/Platform_Wide</td><td>0</td></tr>
<tr><td>S8</td><td></td><td>• Shared_prefs
Nothing
• Sd card</td><td>1</td></tr>
</table>

<table>
<tr><td></td><td></td><td>Recordings from mic stored here (consider the intent filter that allow other apps to start recordings without permission)</td><td></td></tr>
<tr><td>D1</td><td></td><td>No</td><td>0</td></tr>
<tr><td>D2</td><td></td><td>
<table>
<tr><td>Type of component</td><td colspan="2">Hidden</td></tr>
<tr><td></td><td>Explicitly</td><td>Implicitly</td></tr>
<tr><td>Activity</td><td>2</td><td>46</td></tr>
<tr><td>Content Provider</td><td>3</td><td>0</td></tr>
<tr><td>Service</td><td>11</td><td>6</td></tr>
<tr><td>Receiver</td><td>3</td><td>3</td></tr>
<tr><td>Total</td><td>73</td><td>55</td></tr>
</table>
</td><td>55</td></tr>
<tr><td>D3</td><td>Figure 18
Figure 19
Figure 20
Figure 21
Figure 22</td><td>
<table>
<tr><td>Id</td><td>Possible exploit</td><td>Outcome</td></tr>
<tr><td>SE1</td><td>1. Is it possible to start a recording by running the action RECORD_START from an intent?
2. Is it possible to pull the data from the card by using adb and Drozer?</td><td>1. Yes, no perquisite, can start record without signing in, the recording then runs in the background.
2. Can get the data as long its saved on the SD card which seem to be the default choice.

Note:
The user have a choice to save the recordings to the internal memory but its saved on the sdcard by default.</td></tr>
<tr><td></td><td>1. Is it possible to upload or share the recordings</td><td>1. Unclear if the recordings get uploaded but something runs in the background, may be a service</td></tr>
</table>
</td><td>-</td></tr>
</table>

9.1.6 Urbit

Table 13 Urbit

<table>
<tr><th>Test</th><th>Appendix</th><th>Result</th><th>Guideline Conflict</th></tr>
<tr><td>S1</td><td></td><td>
<table>
<tr><td>Type of component</td><td colspan="2">Exported</td></tr>
<tr><td></td><td>No Permission</td><td>Permission</td></tr>
<tr><td>Activity</td><td>1</td><td>0</td></tr>
<tr><td>Content Provider</td><td>0</td><td>0</td></tr>
<tr><td>Service</td><td>(4)firebase</td><td>0</td></tr>
</table>
</td><td>(4)</td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>Receiver</td><td>5</td><td>1</td></tr>
<tr><td>Total</td><td>6+(4)</td><td>1</td></tr>
</table>
The application has no content providers, and the exported items don’t seem to share content with each other so according to the guide lines this is ok. The services with no permissions are firebase classes according to them they don’t need permissions even if they should according to the guide lines.

Attack Surface: 11
No permission: 10</td><td></td></tr>
<tr><td>S2</td><td></td><td>
<table>
<tr><td>Hard ware Uses-Permission</td><td>Usage</td><td>Necessity</td></tr>
<tr><td>ACCESS_FINE_LOCATION</td><td>Location based web shopping</td><td>high</td></tr>
<tr><td>CALL_PHONE</td><td>Shopping</td><td>medium</td></tr>
<tr><td>CAMERA</td><td>Face pic</td><td>Low</td></tr>
<tr><td>GET_ACCOUNTS</td><td>login</td><td>medium</td></tr>
<tr><td>READ_EXTERNAL_STORAGE</td><td>Cache?</td><td>medium</td></tr>
<tr><td>RECEIVE_SMS</td><td>shopping</td><td>high</td></tr>
<tr><td>WRITE_EXTERNAL_STORAGE</td><td>Cache?</td><td>medium</td></tr>
</table>
</td><td>1</td></tr>
<tr><td>S3</td><td></td><td>
<table>
<tr><td>Issue</td><td>Conflict</td></tr>
<tr><td>Pending implicit local intents</td><td>8</td></tr>
<tr><td>Intent no permission</td><td>10</td></tr>
<tr><td>Total</td><td>8</td></tr>
</table>
Test in D3:
action to run
• fuidama.urb_it.activities.SplashActivity
  ○ android.intent.action.VIEW.
• fuidama.urb_it.listener.SmsListener
  ○ fuidama.urb_it.listener.SmsListener record.UploadActivity</td><td>8</td></tr>
<tr><td>S4</td><td></td><td>True</td><td>0</td></tr>
<tr><td>S5</td><td></td><td>False</td><td>0</td></tr>
<tr><td>S6</td><td></td><td>• not obfuscated in billing
• No hardcoded credential</td><td>1</td></tr>
<tr><td>S7</td><td>Figure 14</td><td>• google_analytics_v4.bd
use “CID” to identify user</td><td>0</td></tr>
<tr><td>S8</td><td>Figure 13</td><td>• Shared_prefs
  ○ urbpreferences.xml
Personal information in clear see fig
-Phone
-Name
(-email
-Access token
-Consumerid)

• SD card</td><td>0</td></tr>
</table>

<table>
<tr><td></td><td></td><td>Picasso-cache good!!!</td><td></td></tr>
<tr><td>D1</td><td>Figure 13</td><td>Logs some personal information
Email
Name
This is personal data, the combination of Email address and name is unique.
(under some circumstance there could be two or more individuals using the same email with the same name, but that’s very unlikely).</td><td>1</td></tr>
<tr><td>D2</td><td></td><td><table><tr><td rowspan="2">Type of component</td><td colspan="2">Hidden</td></tr><tr><td>Explicitly</td><td>Implicitly</td></tr><tr><td>Activity</td><td>1</td><td>36</td></tr><tr><td>Content Provider</td><td>1</td><td>0</td></tr><tr><td>Service</td><td>2</td><td>0</td></tr><tr><td>Receiver</td><td>1</td><td>0</td></tr><tr><td>Total</td><td>5</td><td>36</td></tr></table></td><td>36</td></tr>
<tr><td>D3</td><td></td><td><table><tr><td>Id</td><td>Possible exploit</td><td>Outcome</td></tr><tr><td>SE1</td><td>1. Is it possible to run activity to get passed login what kind of id is it?</td><td>1. No, end up in login activity anyway</td></tr><tr><td>SE2</td><td>1. Is it possible to send broadcast through action: android.provider. Telephony. SMS_RECEIVED</td><td>1. No Permission Denial: not allowed to send broadcast android. provider.Telephony. SMS_RECEIVED from pid=3759, uid=10057</td></tr></table></td><td>-</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table>

9.1.7 Rinkside 3

Table 14 Rinkside 3

<table>
<tr><th>Test</th><th>Appendix</th><th>Result</th><th>Guideline Conflict</th></tr>
<tr><td>S1</td><td></td><td><table><tr><td rowspan="2">Type of component</td><td colspan="2">Exported</td></tr><tr><td>No Permission</td><td>Permission</td></tr><tr><td>Activity</td><td>3</td><td>0</td></tr><tr><td>Content Provider</td><td>0</td><td>0</td></tr><tr><td>Service</td><td>0</td><td>0</td></tr></table></td><td>0</td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>Receiver</td><td>1</td><td>1</td></tr>
<tr><td>Total</td><td>4</td><td>0</td></tr>
</table>
The application has no content providers, and the exported items don’t seem to share content with each other so according to the guide lines this is ok.
The attack surface is rather small.

Attack Surface: 5
No permissions: 4</td><td></td></tr>
<tr><td>S2</td><td></td><td>
<table>
<tr><td>Hard ware Uses-Permission</td><td>Necessity</td></tr>
<tr><td>WRITE_EXTERNAL_STORAGE</td><td>medium</td></tr>
<tr><td>READ_EXTERNAL_STORAGE</td><td>medium</td></tr>
</table>
</td><td>0</td></tr>
<tr><td>S3</td><td></td><td>
<table>
<tr><td>Issue</td><td>Conflict</td></tr>
<tr><td>Pending implicit local intents</td><td>3</td></tr>
<tr><td>Intent no permission</td><td>4</td></tr>
<tr><td>Total</td><td>7</td></tr>
</table>
Test in D3:
activity to run
• com.adobe.phonegap.push.PushHandlerActivity</td><td>3</td></tr>
<tr><td>S4</td><td></td><td>Default</td><td>1</td></tr>
<tr><td>S5</td><td></td><td>False</td><td>0</td></tr>
<tr><td>S6</td><td></td><td>• No credit card so no need for obfuscation?
• Couldn’t find any of the code words hardcoded
• No credit cards handled</td><td>0</td></tr>
<tr><td>S7</td><td></td><td>• google_conversion_tracking.db
Database is empty don’t know if encrypted, tracks conversations so can be a privacy concern.</td><td>0</td></tr>
<tr><td>S8</td><td></td><td>• Shared_prefs
com.facebook.sdk.appEventPreferences.xml see fig
com.google.android.gms.appid.xml properly encrypted!!!
Nothing interesting!!!!
• SD card
Cache on SD card don’t seem to be much of a problem, no personal information, (empty though!!!!!!!)</td><td>0</td></tr>
<tr><td>D1</td><td></td><td>• Login
Nothing
• Search
Nothing</td><td>0</td></tr>
<tr><td>D2</td><td></td><td>
<table>
<tr><td rowspan="2">Type of component</td><td colspan="2">Hidden</td></tr>
<tr><td>Explicitly</td><td>Implicitly</td></tr>
<tr><td>Activity</td><td>1</td><td>1</td></tr>
<tr><td>Content Provider</td><td>0</td><td>0</td></tr>
<tr><td>Service</td><td>3</td><td>0</td></tr>
<tr><td>Receiver</td><td>0</td><td>1</td></tr>
</table>
</td><td>2</td></tr>
</table>

		Total: 4 / 2	
D3	Table 1 Figure 36	Id: SE1 — Possible exploit: 1. Is it possible to start activity from an intent? com.adobe.phonegap.push.PushHandlerActivity — Outcome: 1. No, But the application crashes	-

9.1.8 Pinterest

Table 15 Pinterest

Test	Appendix	Result	*Guideline Conflict*
S1		Type of component / Exported: No Permission / Permission Activity: 6 / 0 Content Provider: 0 / 0 Service: 4 / 1 Receiver: 3 / 2 Total: 13 / 3 The application has no content providers, and the exported items don't seem to share content with each other so according to the guide lines this is ok. Attack Surface: 16 No permission: 13	0
S2		Hard ware Uses-Permission / Usage / Necessity CAMERA / Face pic / low READ_CONTACTS / Linking social accounts / medium GET_ACCOUNTS / login / medium ACCESS_FINE_LOCATION / Advertisement? / low READ_EXTERNAL_STORAGE / Cache? / medium WRITE_EXTERNAL_STORAGE / Cache? / medium	2

S3	Figure 28	Permission: null
		Pending implicit local intents: 1
		Intent no permission: 13
		Total: 17
		
		Test in D3:
		• com.pinterest.sdk.PinterestOauthActivity
		• com.pinterest.activity.create.PinItActivity
		  ○ com.pinterest.action.PIN_IT
		  ○ android.intent.action.SEND
			1

S4		False	0
S5		False	0

<table>
<tr><td>S6</td><td></td><td>1. Obfuscated in parts of code where credentials are handled
2. No hardcoded information found</td><td>0</td></tr>
<tr><td>S7</td><td></td><td>Nothing saved here!</td><td>0</td></tr>
<tr><td>S8</td><td>Figure 27</td><td>1. Shared_prefs
Basically nothing, found username in clear see fig

2. SD card
Can’t find anything on the SD card</td><td>0</td></tr>
<tr><td>D1</td><td></td><td>1. login
nothing
2. searches
nothing</td><td>0</td></tr>
<tr><td>D2</td><td></td><td>
<table>
<tr><td>Type of component</td><td colspan="2">Hidden</td></tr>
<tr><td></td><td>Explicitly</td><td>Implicitly</td></tr>
<tr><td>Activity</td><td>2</td><td>24</td></tr>
<tr><td>Content Provider</td><td>4</td><td>0</td></tr>
<tr><td>Service</td><td>4</td><td>3</td></tr>
<tr><td>Receiver</td><td>0</td><td>5</td></tr>
<tr><td>Total</td><td>42</td><td>32</td></tr>
</table>
</td><td>32</td></tr>
<tr><td>D3</td><td>Figure 31</td><td>
<table>
<tr><td>Id</td><td>Possible exploit</td><td>Outcome</td></tr>
<tr><td>SE1</td><td>1. What is com.pinterest.sdk.PinterestOauthActivity</td><td>1. Need data seems like try to load login page? See fig</td></tr>
<tr><td>SE2</td><td>1. Is it possible to put up new pins from Drozer by com.pinterest.action.PIN_IT
2. Data:
(type: text/plain)
(type: image/*)
Don’t know if it’s just the data that’s malformed or some sort of permission is needed</td><td>1. Reloads the page, need data?
Checking the log:

pid6358
Subscriber to unregister was not registered before.</td></tr>
<tr><td>S3</td><td>1. Is it possible to put up new pins from Drozer by com.pinterest.action.SEND
2. Data:
(type: text/plain)
(type: image/*)
Don’t know if it’s just the data that’s malformed or</td><td>1. Checking the log find message:

never saw a connection for the pid: 6358</td></tr>
</table>
</td><td>-</td></tr>
</table>

			some sort of permission is needed?		

9.1.9 Postnord

Table 16 Postnord

<table>
<tr><th>Test</th><th>Appendix</th><th>Result</th><th>Guideline conflict</th></tr>
<tr><td>S1</td><td></td><td>
<table>
<tr><td rowspan="2">Type of component</td><td colspan="2">Exported</td></tr>
<tr><td>No Permission</td><td>Permission</td></tr>
<tr><td>Activity</td><td>2</td><td>0</td></tr>
<tr><td>Content Provider</td><td>0</td><td>0</td></tr>
<tr><td>Service</td><td>5</td><td>1</td></tr>
<tr><td>Receiver</td><td>6</td><td>1</td></tr>
<tr><td>Total</td><td>13</td><td>2</td></tr>
</table>
The application has no content providers, and the exported items don’t seem to share content with other apps so according to the guide lines this is ok.

Attack Surface: 15
No permission: 13
</td><td>0</td></tr>
<tr><td>S2</td><td></td><td>
<table>
<tr><td>Hard ware Uses-Permission</td><td>Usage</td><td>Necessity</td></tr>
<tr><td>CAMERA</td><td>Scan parcels</td><td>medium</td></tr>
<tr><td>RECEIVE_SMS</td><td>Receiving packets</td><td>high</td></tr>
<tr><td>GET_ACCOUNTS</td><td>login</td><td>medium</td></tr>
<tr><td>ACCESS_COARSE_LOCATION</td><td>adapt</td><td>high</td></tr>
<tr><td>ACCESS_FINE_LOCATION</td><td>Find closest service point</td><td>high</td></tr>
<tr><td>READ_EXTERNAL_STORAGE</td><td>Don’t know</td><td>?</td></tr>
<tr><td>WRITE_EXTERNAL_STORAGE</td><td>Don’t know</td><td>?</td></tr>
</table>
All permissions seem to be relevant for the core function. Location is needed to for the map function to find nearest service point. Get accounts is used to login. The camera is used
to scan packet codes. It's unclear what read/write external do.
</td><td>0</td></tr>
<tr><td>S3</td><td></td><td>
<table>
<tr><td>Issue</td><td>null</td></tr>
<tr><td>Pending implicit local intents</td><td>4</td></tr>
<tr><td>Intent no permission</td><td>18</td></tr>
<tr><td>Total</td><td>22</td></tr>
</table>
</td><td>4</td></tr>
</table>

<table>
<tr><td></td><td></td><td>Test in D3:
To run
android.provider.Telephony.SMS_RECEIVED</td><td></td></tr>
<tr><td>S4</td><td></td><td>True</td><td>0</td></tr>
<tr><td>S5</td><td></td><td>False</td><td>0</td></tr>
<tr><td>S6</td><td></td><td>1. Obfuscated in parts of code where credentials are handled
2. No hardcoded information found
3. Uses MD5 which is considered insecure</td><td>1</td></tr>
<tr><td>S7</td><td></td><td>• Device.db
Contains information about parcels in cleartext may be a bad idea</td><td>0</td></tr>
<tr><td>S8</td><td>Figure 25</td><td>1. Shared_prefs
Some personal information in clear
2. SD card
Nothing !!!!
why permission to write and read??????</td><td>0</td></tr>
<tr><td>D1</td><td>Figure 26</td><td>1. Searching for parcels
Shows information in clear:
Search Packet number(not personal information)</td><td>0</td></tr>
<tr><td>D2</td><td></td><td>
<table>
<tr><td rowspan="2">Type of component</td><td colspan="2">Hidden</td></tr>
<tr><td>Explicitly</td><td>Implicitly</td></tr>
<tr><td>Activity</td><td>1</td><td>9</td></tr>
<tr><td>Content Provider</td><td>2</td><td>0</td></tr>
<tr><td>Service</td><td>3</td><td>2</td></tr>
<tr><td>Receiver</td><td>2</td><td>3</td></tr>
<tr><td>Total</td><td>8</td><td>14</td></tr>
</table>
</td><td>14</td></tr>
<tr><td>D3</td><td>Table 23</td><td>
<table>
<tr><td>Id</td><td>Possible exploit</td><td>Outcome</td></tr>
<tr><td>SE1</td><td>1. Try to send a sms received broadcast</td><td>1. Permission Denial: not allowed to send broadcast android.provider. Telephony.SMS_ RECEIVED from pid=4711, uid=10057
Seems like the broadcast have to come from the telephony’s uid to get accepted</td></tr>
</table>
</td><td></td></tr>
</table>

9.1.10 Tempelrun

Table 17 Tempelrun

<table>
<tr><th>Test</th><th>Appendix</th><th>Result</th><th>Guideline Conflict</th></tr>
<tr><td>S1</td><td></td><td>
<table>
<tr><td rowspan="2">Type of component</td><td colspan="2">Exported</td></tr>
<tr><td>No Permission</td><td>Permission</td></tr>
<tr><td>Activity</td><td>2</td><td>0</td></tr>
<tr><td>Content Provider</td><td>0</td><td>0</td></tr>
<tr><td>Service</td><td>0</td><td>0</td></tr>
<tr><td>Receiver</td><td>2</td><td>0</td></tr>
<tr><td>Total</td><td>4</td><td>0</td></tr>
</table>
The application has no content providers, and the exported items don’t seem to share content with each other so according to the guide lines this is ok.

Attack Surface: 2
No permission: 2
</td><td>0</td></tr>
<tr><td>S2</td><td></td><td>
<table>
<tr><td>Hard ware Uses-Permission</td><td>Usage</td><td>Necessity</td></tr>
<tr><td>READ_EXTERNAL_STORAGE</td><td>Save recordings of gameplay</td><td>Medium</td></tr>
<tr><td>WRITE_EXTERNAL_STORAGE</td><td>Save recordings of gameplay</td><td>Medium</td></tr>
</table>
</td><td>0</td></tr>
<tr><td>S3</td><td>Figure 37</td><td>
<table>
<tr><td>Issue</td><td>null</td></tr>
<tr><td>Pending implicit local intents</td><td>4</td></tr>
<tr><td>Intent no permission</td><td>4</td></tr>
<tr><td>Total</td><td>8</td></tr>
</table>
Test in D3:
None of the suggested tests applies
Run an activity and see what happens
• Component.com. imangi.templerun com.flurry. android.CatalogActivity
</td><td>4</td></tr>
<tr><td>S4</td><td></td><td>Default</td><td>1</td></tr>
<tr><td>S5</td><td></td><td>False</td><td>0</td></tr>
<tr><td>S6</td><td>Figure 35
Figure 33</td><td>1. obfuscated in billing part
2. No hardcoded information found
3. public class DeviceManager contains methods to get the hardware deviceid but thers no permission to get the information.
4. Found a method called isHacked which always returns false value</td><td>0</td></tr>
</table>

<table>
<tr><td>S7</td><td></td><td>Can’t find any database</td><td>0</td></tr>
<tr><td>S8</td><td></td><td>1. Shared_prefs
Nothing,,some xmls
2. SD card
Save recordings of gameplay</td><td>0</td></tr>
<tr><td>D1</td><td>Figure 32 Can’t get IMEI, no permission (Templerun)Figure 32</td><td>1. IMEI number in log
(even though there’s no permission to get the number this kind of information should not be in the system log)</td><td>1</td></tr>
<tr><td>D2</td><td></td><td>
<table>
<tr><td rowspan="2">Type of component</td><td colspan="2">Hidden</td></tr>
<tr><td>Explicitly</td><td>Implicitly</td></tr>
<tr><td>Activity</td><td>0</td><td>18</td></tr>
<tr><td>Content Provider</td><td>0</td><td>0</td></tr>
<tr><td>Service</td><td>1</td><td>0</td></tr>
<tr><td>Receiver</td><td>0</td><td>0</td></tr>
<tr><td>Total</td><td>1</td><td>18</td></tr>
</table>
According to the guide lines for core quality all these items should be exported explicitly.
</td><td>18</td></tr>
<tr><td>D3</td><td>Table 25
Figure 36 D3 test Application crash when intent is sent (Templerun)</td><td>
<table>
<tr><td>Id</td><td>Possible exploit</td><td>Outcome</td></tr>
<tr><td>SE1</td><td>1. Start activity by intent and see what happens</td><td>1. The application crashes</td></tr>
</table>
</td><td></td></tr>
</table>

9.2 Test Figures and Tables

9.2.1 wish

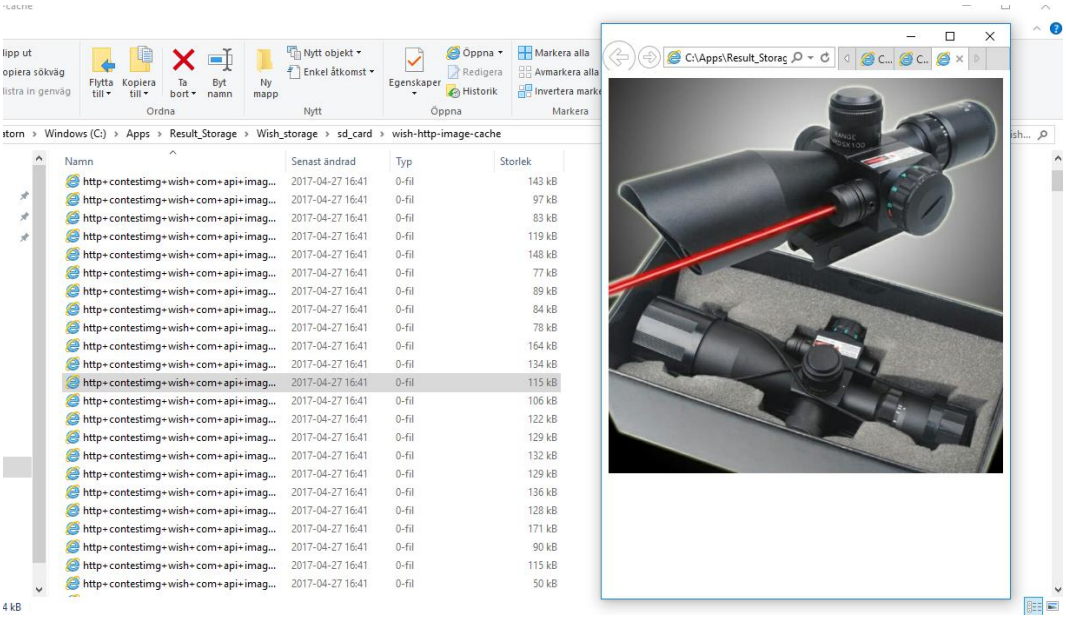

Figure 5 Search cache in the SD card may be a privacy concern (Wish)

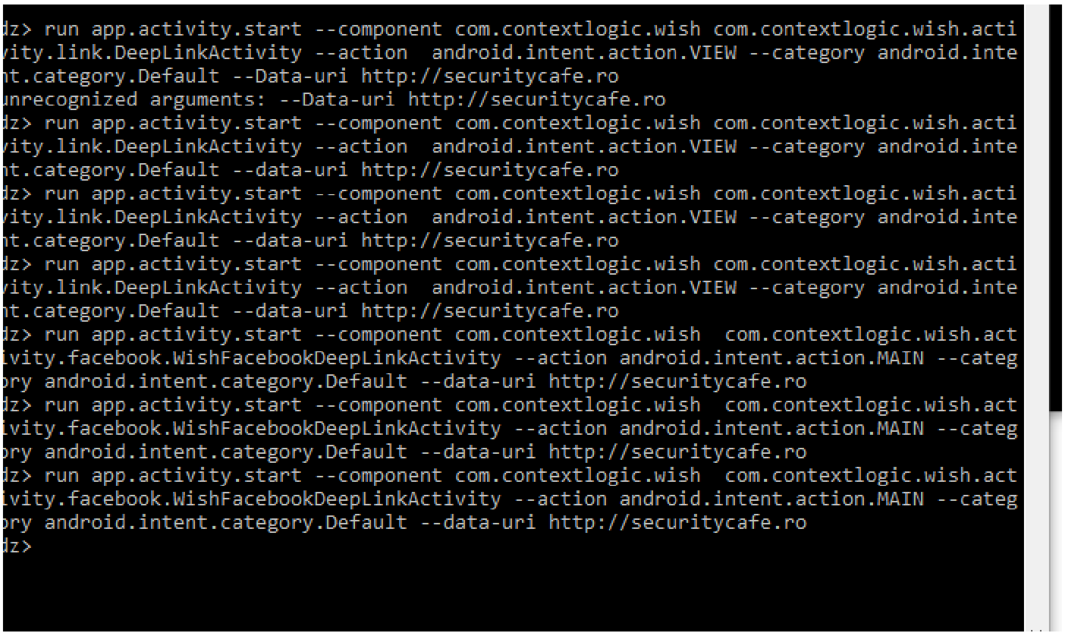

Figure 6 running intents to test vulnerabilities (Wish)

Table 18 D3 tests (Wish)

Nr	D3 tests in Drozer
1	run app.broadcast.send --component com.contextlogic.wish com.contextlogic.wish.receiver.DeviceIdReceiver --action com.contextlogic.wish.DEVICE_ID
2	run app.activity.start --component com.contextlogic.wish com.contextlogic.wish.activity.link.DeepLinkActivity --action android.intent.action.VIEW --data-uri http://google.se
3	run app.activity.start --component com.contextlogic.wish com.contextlogic.wish.activity.facebook.WishFacebookDeepLinkActivity --action android.intent.action.VIEW --data-uri http://google.se

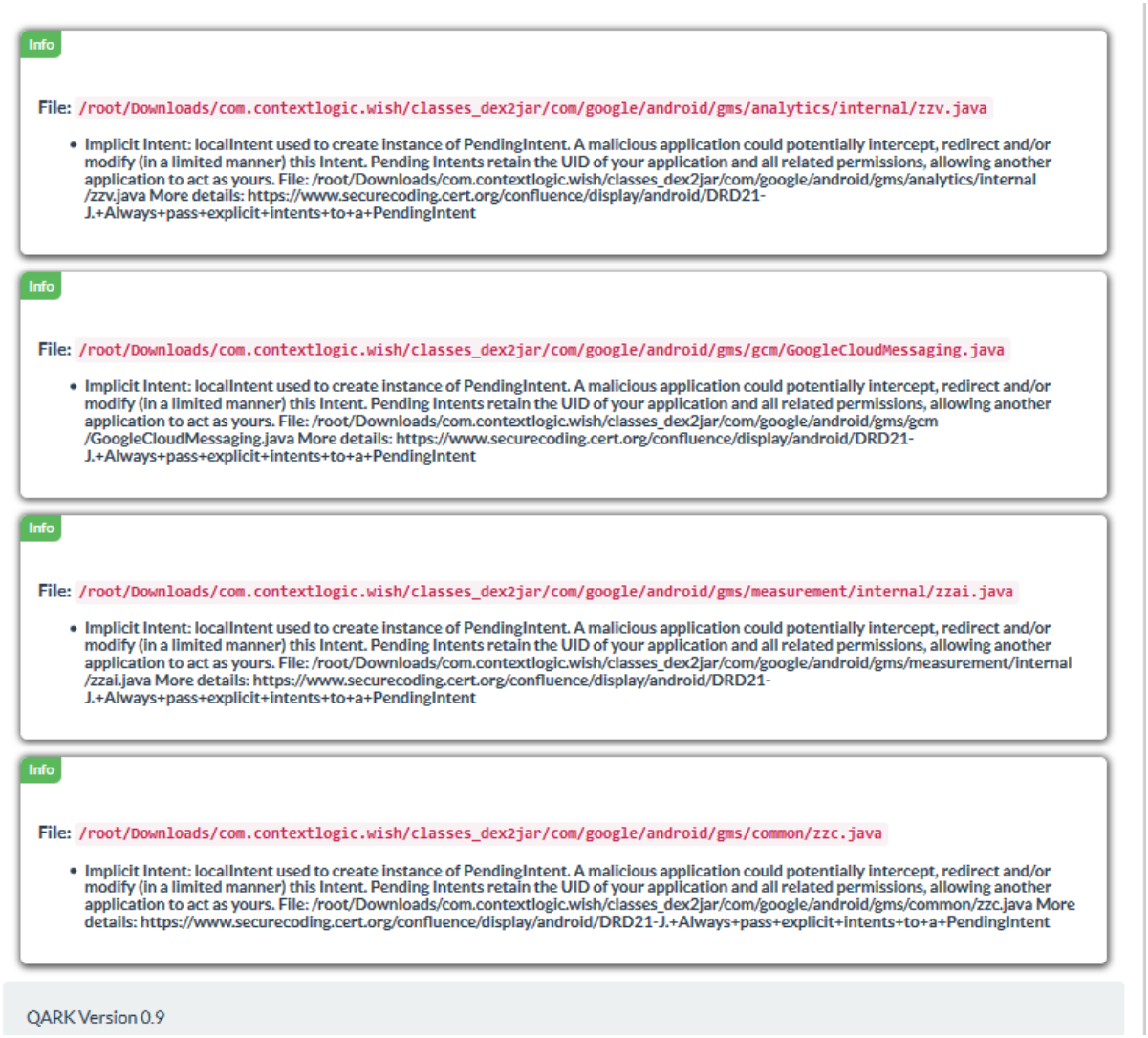

Figure 7 Pending local intents (Wish)

Error! Reference source not found.

9.2.2 Eniro

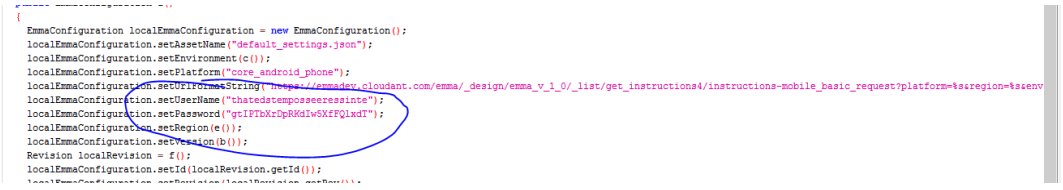

Figure 8 Password and username to webpage hardcoded (Eniro)

Figure 9 Pending intents in Qark (Eniro)

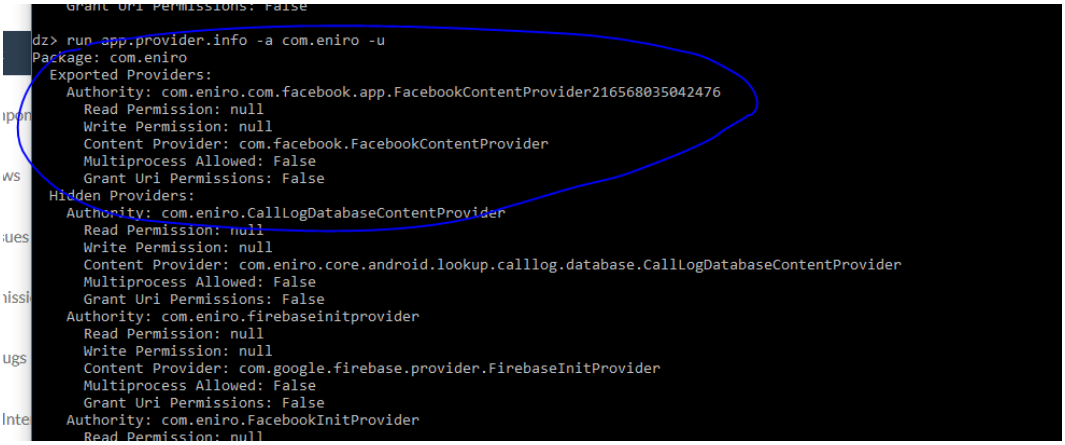

Figure 10 Content provider in Drozer with null permission can't be found in manifest (Eniro)

Table 19 D3 Test (Eniro)

Nr	D3 tests in Drozer
1	run app.activity.start --component com.eniro com.eniro.core.android.map.MapActivity --action android.intent.action.VIEW --category android.intent.category.DEFAULT --data-uri http://google.se

9.2.3 WWMobile

Figure 11 Pending local intents (WWMobile)

Table 20 D3 Test (WWMobile)

Nr	D3 tests in Drozer
1	run app.activity.start --component com.weightwatchers.mobile com.weightwatchers.community.ui.activity.UserProfileActivity --action android.intent.action.VIEW --category android.intent.category.DEFAULT --data-uri http://google.se
2	run app.activity.start --component com.weightwatchers.mobile com.weightwatchers.community.ui.activity.NewPostActivity --action com.weightwatchers.food.action.SEARCH --category android.intent.category.DEFAULT --data-uri http://google.se
3	run app.activity.start --component com.weightwatchers.mobile com.weightwatchers.mobile.ui.activity.MainSearchActivity --action android.intent.action.VIEW --category android.intent.category.DEFAULT --data-uri http://google.se

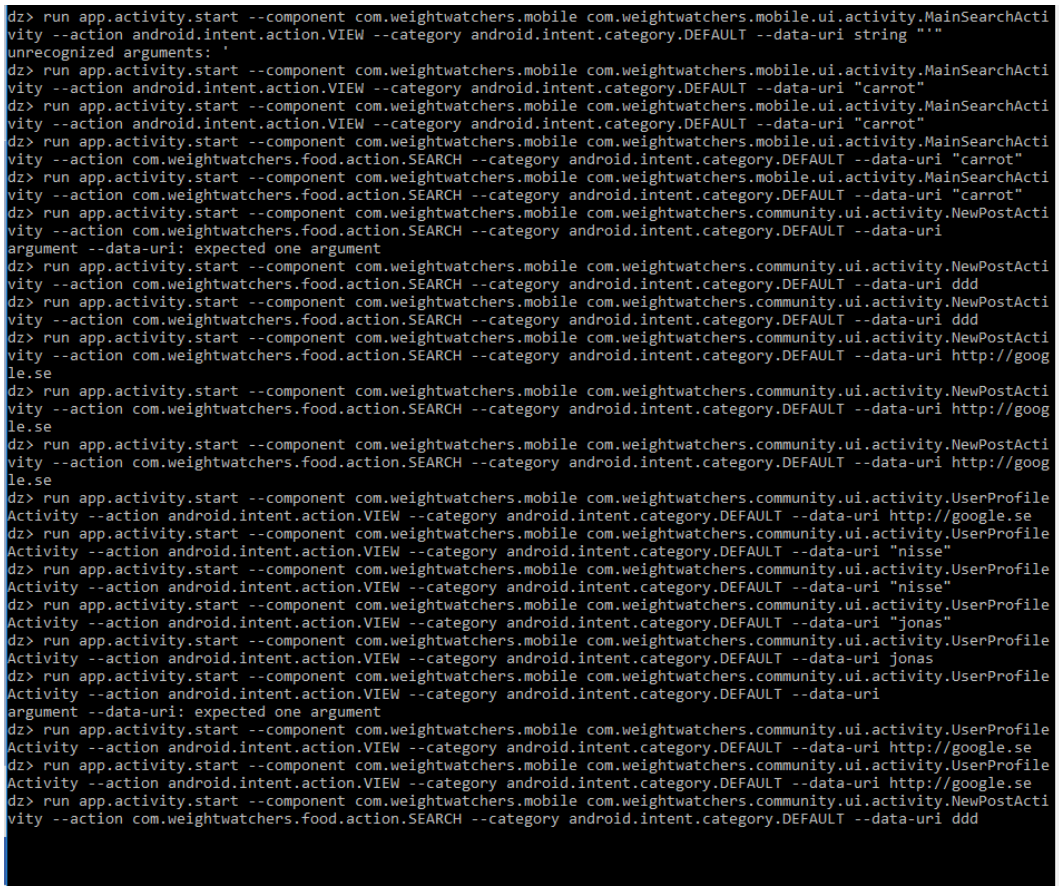

Figure 12 Varius intents tested parallel with d3 test (WWMobile)

Table 10 WWMobile

9.2.4 Urbit

Table 21 D3 test (Urbit)

Nr	D3 tests in Drozer
1	run app.activity.start --component fuidama.urb_it fuidama.urb_it.activities.SplashActivity --action android.intent.action.VIEW --category android.intent.category.DEFAULT
2	run app.activity.start --component fuidama.urb_it fuidama.urb_it.listener.SmsListener --action android.provider.Telephony.SMS_RECEIVED --category android.intent.category.DEFAULT

Figure 13 Personal information name, consumer id (Urbit)

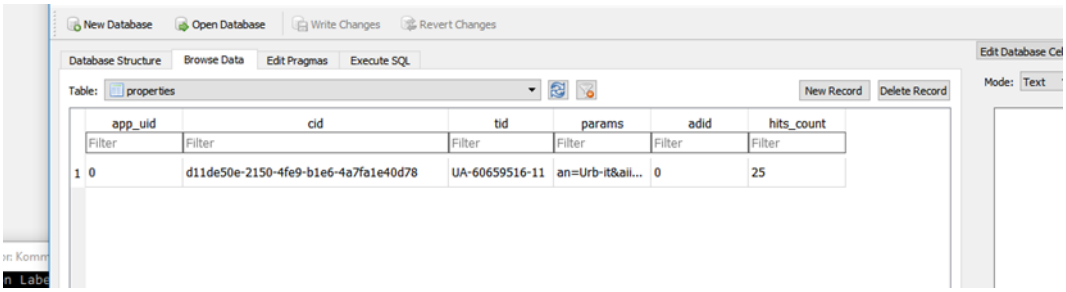

Figure 14 Cid saved on disk, encrypted??(Urbit)

Figure 15 Personal information (phone number and name) in clear b (Urbit)

9.2.5 Bortskankes

Nothing!!

9.2.6 Soundcloud

Table 22 D3 Testing intents (Soundcloud)

Nr	D3 tests in Drozer

1	run app.activity.start --component com.soundcloud.android com.soundcloud.android.creators.record.RecordActivity --action com.soundcloud.android.action.RECORD_START --category android.intent.category.DEFAULT
2	run app.activity.start --component com.soundcloud.android com.soundcloud.android.creators.record.UploadActivity --action android.intent.action.SEND --category android.intent.category.DEFAULT
3	run app.activity.start --component com.soundcloud.android com.soundcloud.android.creators.record.UploadActivity --action com.soundcloud.android.SHARE --category android.intent.category.DEFAULT

Figure 16 Pending local intent (Soundcloud)

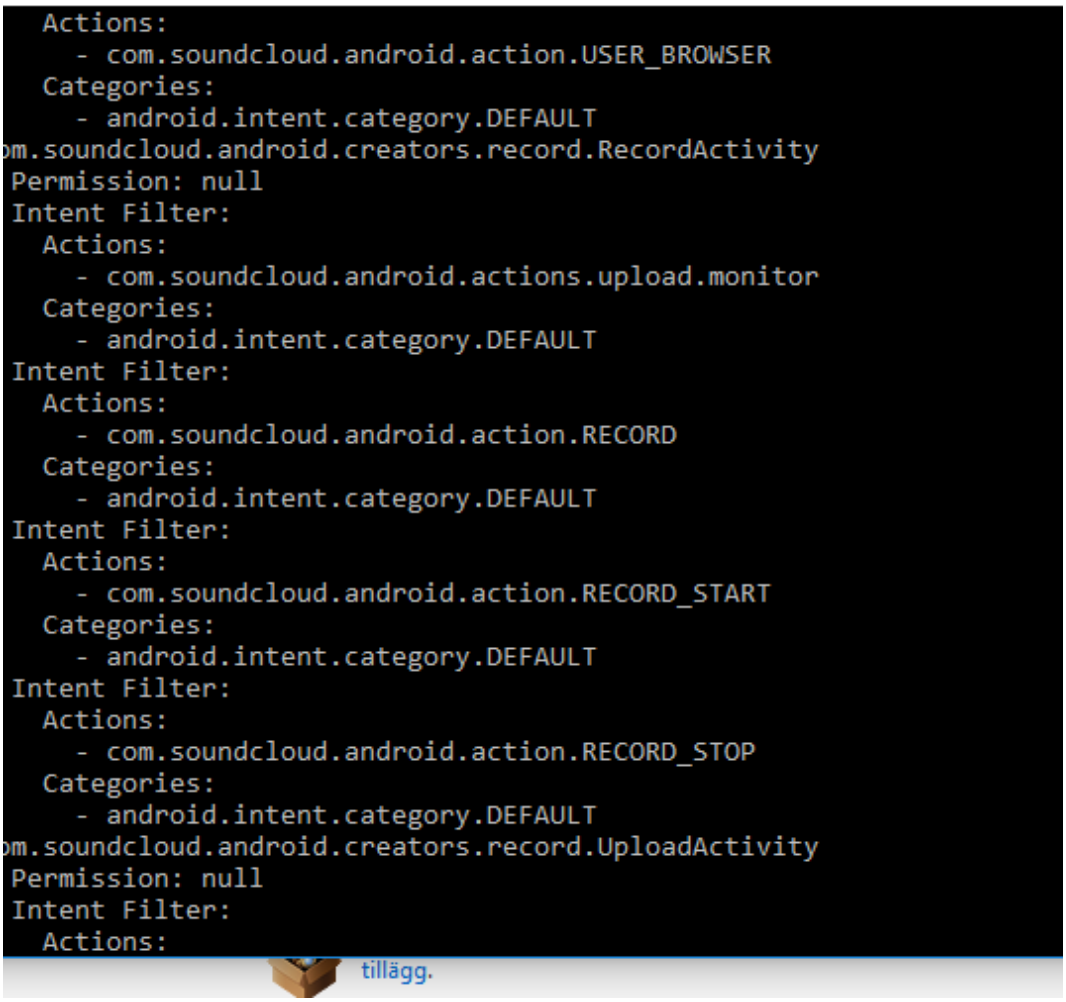

Figure 17 Intent filters actions for recording (Soundcloud)

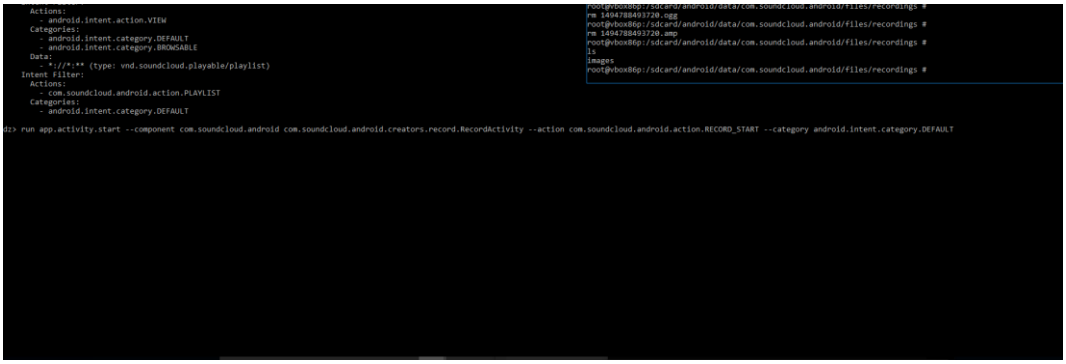

Figure 18 (Right upper corner) SD card empty before running intent (Soundcloud)

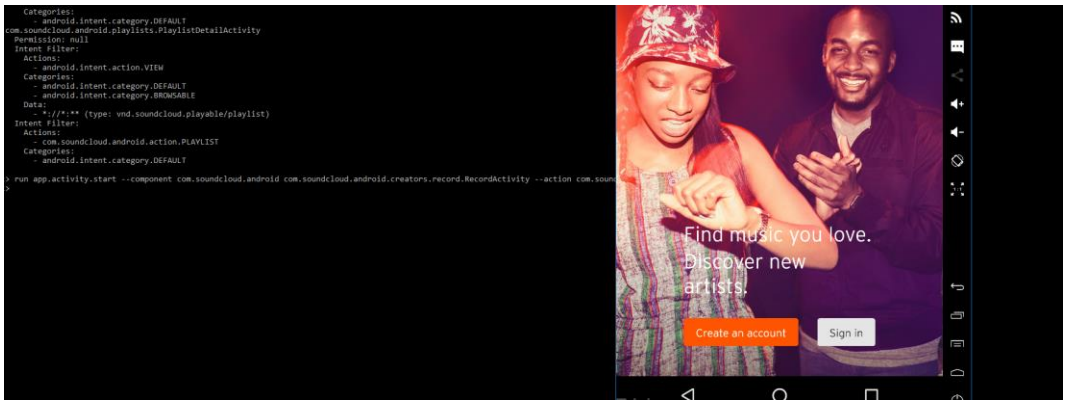

Figure 19 Running intent to start recording, showing signing activity, recording has started in background (Soundcloud)

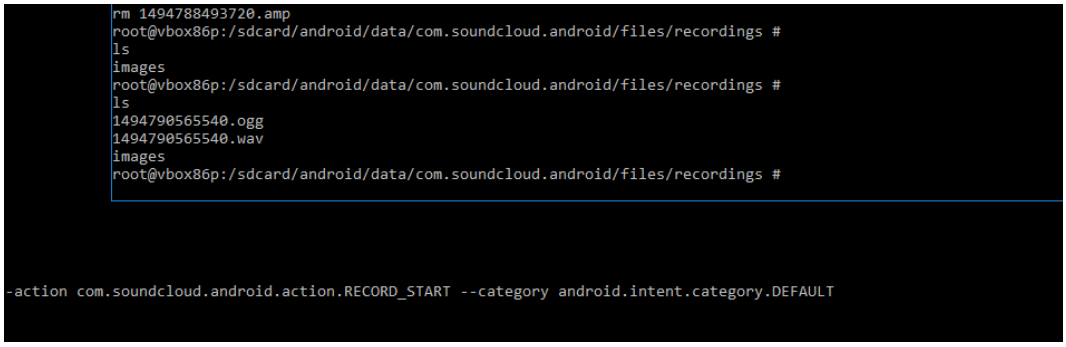

Figure 20 New audio files on SD card after recording started (Soundcloud)

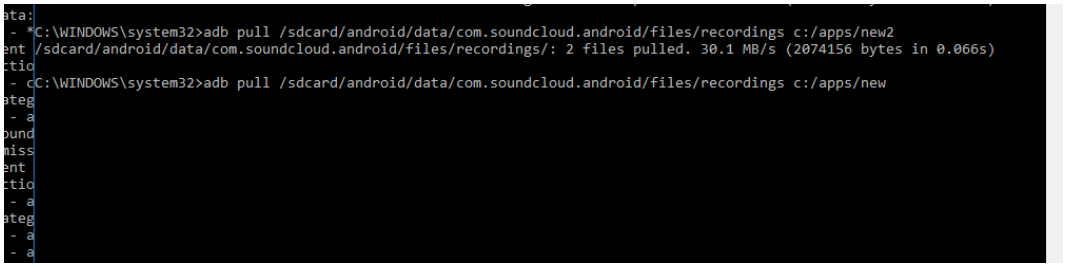

Figure 21 Pulling the audio files from SD card with adb (Soundcloud)

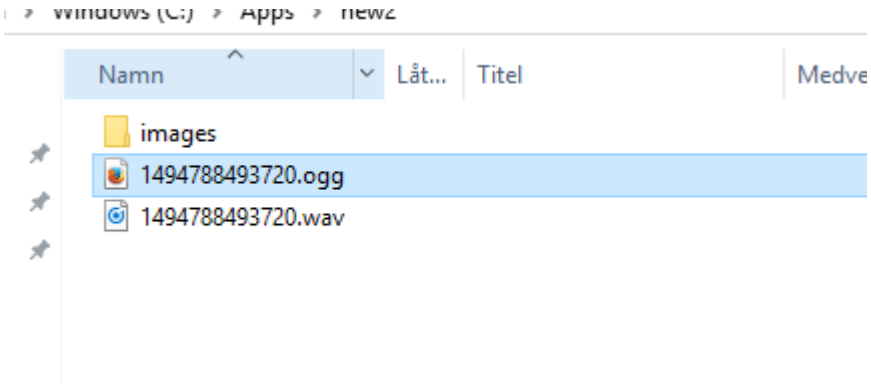

Figure 22 Audio files copied from device (Soundcloud)

9.2.7 Rinkside 3

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<map>
    <string name="anonymousAppDeviceGUID">XZ9da50a3e-4402-4ad9-aa54-49239e69fdf4</string>
</map>
```

Figure 23 Advertisement id? (Rinkside3)

Table 1 D3 Test (Rinkside3)

Nr	D3 tests in Drozer
1	run app.activity.start --component se.passionlab.rinkside3 com.adobe.phonegap.push. PushHandlerActivity

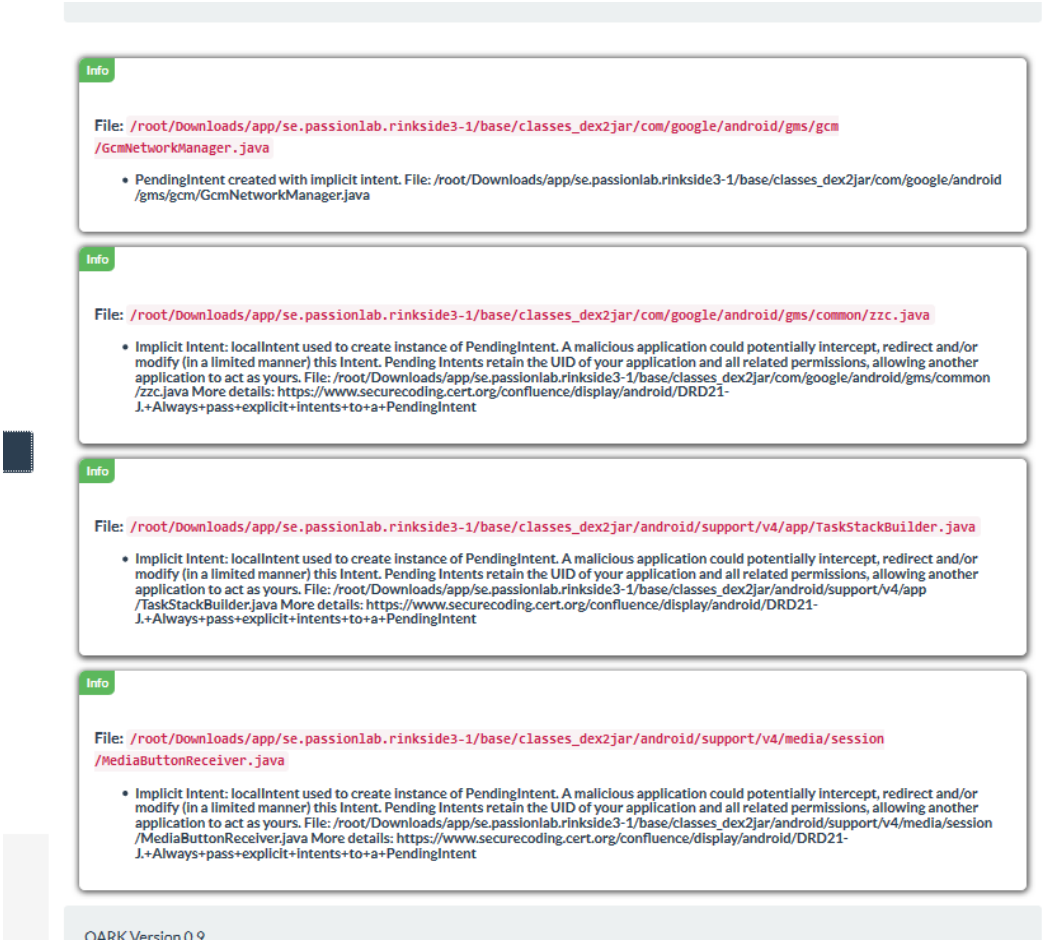

Figure 1Pending local intents (Rinkside3)

Figure 24 Pending Intent (Rinkside3)

9.2.8 Postnord

```
<boolean value="true" name="navigation_drawer_learned"/>
<string name="deleted_items"/>
<boolean value="false" name="feature_retired_beacons"/>
<boolean value="false" name="feature_retired_digitalSlip"/>
<boolean value="false" name="feature_retired_getParcels"/>
<boolean value="false" name="showed_feature_message_digitalSlip"/>
<int value="0" name="sms_notification_number"/>
<float value="0.0" name="1_66666666666666"/>
/map>
```

Figure 25 Shared_prefs nothing? (Postnord)

Figure 26 Parcel in system log when searching (Postnord)

Table 23 D3 test (Postnord)

Nr	D3 tests in Drozer
1	run app.broadcast.send --component se.postnord.private com.postnord.MaCSMSBroadcastReceiver --action android.provider.Telephony.SMS_RECEIVED

9.2.9 Pintrerest

Figure 27 Personal information(username) in shared prefs (Pinterest)

Table 24 D3 tests (pinterest)

Nr	D3 tests in Drozer
1	run app.activity.start --component com.pinterest com.pinterest.sdk. PinterestOauthActivity
2	run app.activity.start --component com.pinterest com.pinterest.activity.create.PinItActivity --action com.pinterest.action.PIN_IT --category android.intent.category.DEFAULT
3	run app.activity.start --component com.pinterest com.pinterest.activity.create.PinItActivity --action android.intent.action.SEND --category android.intent.category.DEFAULT

Figure 28 *Pending local intents (Pinterest)*

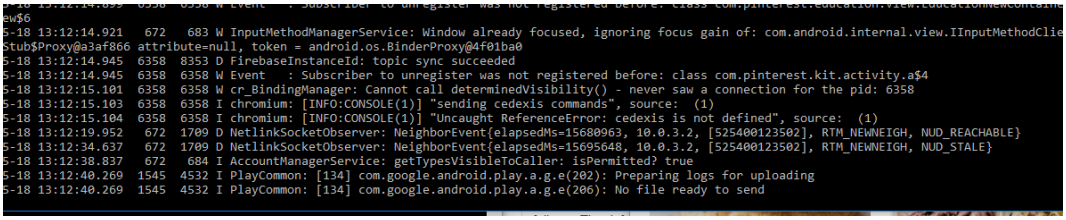

Figure 29 Logcat when running SE2 (Pinterest)

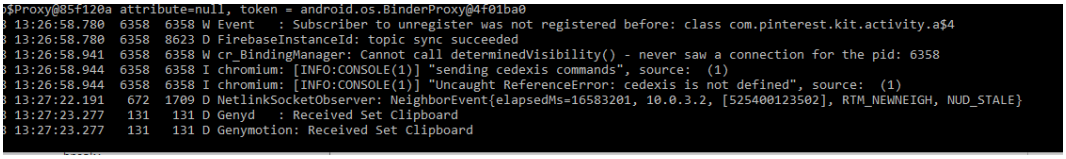

Figure 30 Logcat when running SE3 (Pinterest)

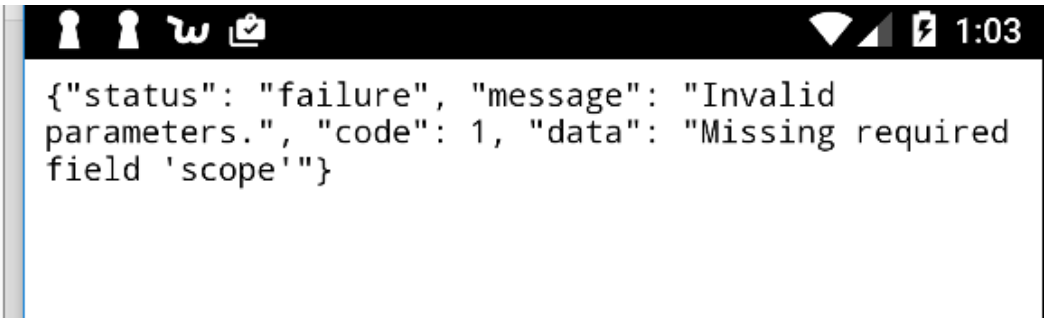

Figure 31 Trying to start activity (Pinterest)

9.2.10 Tempelrun

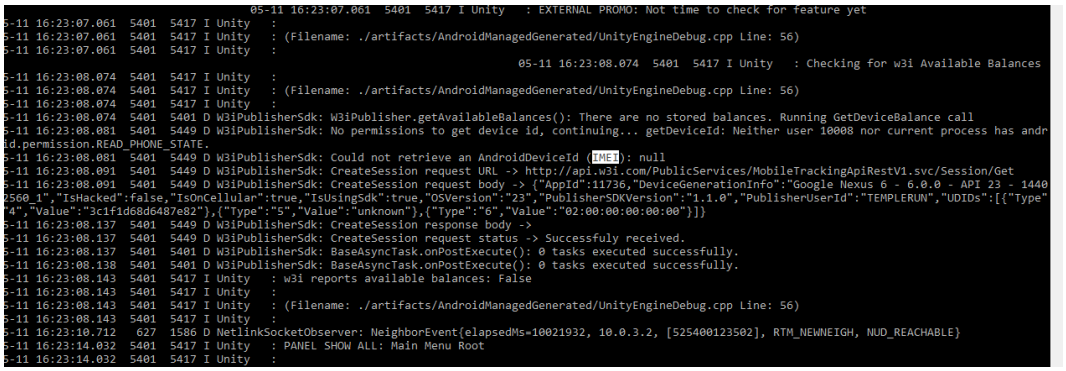

Figure 32 Can't get IMEI, no permission (Templerun)

```
{
  private static Device device;
  private static DeviceManager deviceManager;
  private final boolean IS_CHECKING_ROOTED = false;
  private Context context;

  private DeviceManager(Context paramContext)
  {
    this.context = paramContext;
  }

  private String getAndroidDeviceId()
  {
    try
    {
      String str = ((TelephonyManager)this.context.getSystemService("phone")).getDeviceId();
      return str;
    }
    catch (SecurityException localSecurityException)
    {
      Log.d("No permissions to get device id, continuing... " + localSecurityException.getMessage());
    }
    return null;
  }

  private String getAndroidId()
  {
    try
    {
      String str = Settings.Secure.getString(this.context.getContentResolver(), "android_id");
      return str;
    }
    catch (Exception localException)
    {
      Log.d("DeviceManager: Unexpected exception caught in getAndroidId()");
      localException.printStackTrace();
    }
    return null;
```

Figure 33 Trying to get IMEI (Templerun)

```
52  {
53    try
54    {
55      String str = SerialNumberFactory.getSerialNumber();
56      return str;
57    }
58    catch (Exception localException)
59    {
60      Log.d("DeviceManager: Unexpected exception caught in getAndroidSerialNumber()");
61      localException.printStackTrace();
62    }
63    return null;
64  }
65
66  private String getDeviceId()
67  {
68    String str = getAndroidDeviceId();
69    if (str != null)
70    {
71      Log.d("Found an AndroidDeviceId (IMEI): " + str);
72      return str;
73    }
74    Log.d("Could not retrieve an AndroidDeviceId (IMEI): " + str);
75    return str;
76  }
77
78  public static Device getDeviceInstance(Context paramContext)
79  {
80    if (paramContext == null) {}
81    for (;;)
82    {
83      try
84      {
85        paramContext = AdvertiserSharedData.getContext();
86        if (deviceManager == null)
87        {
```

Figure 34 Method getting the IMEI (Templerun)

```
140     }
141     return null;
142   }
143
144   private boolean isHacked()
145   {
146     return false;
147   }
148
```

Figure 35 Strange method, it always returns false (Templerun)

Table 25 D3 test (Templerun)

Nr	D3 tests in Drozer
1	D3 tests in Drozer run app.activity.start --component com.imangi.templerun com.flurry.android.CatalogActivity

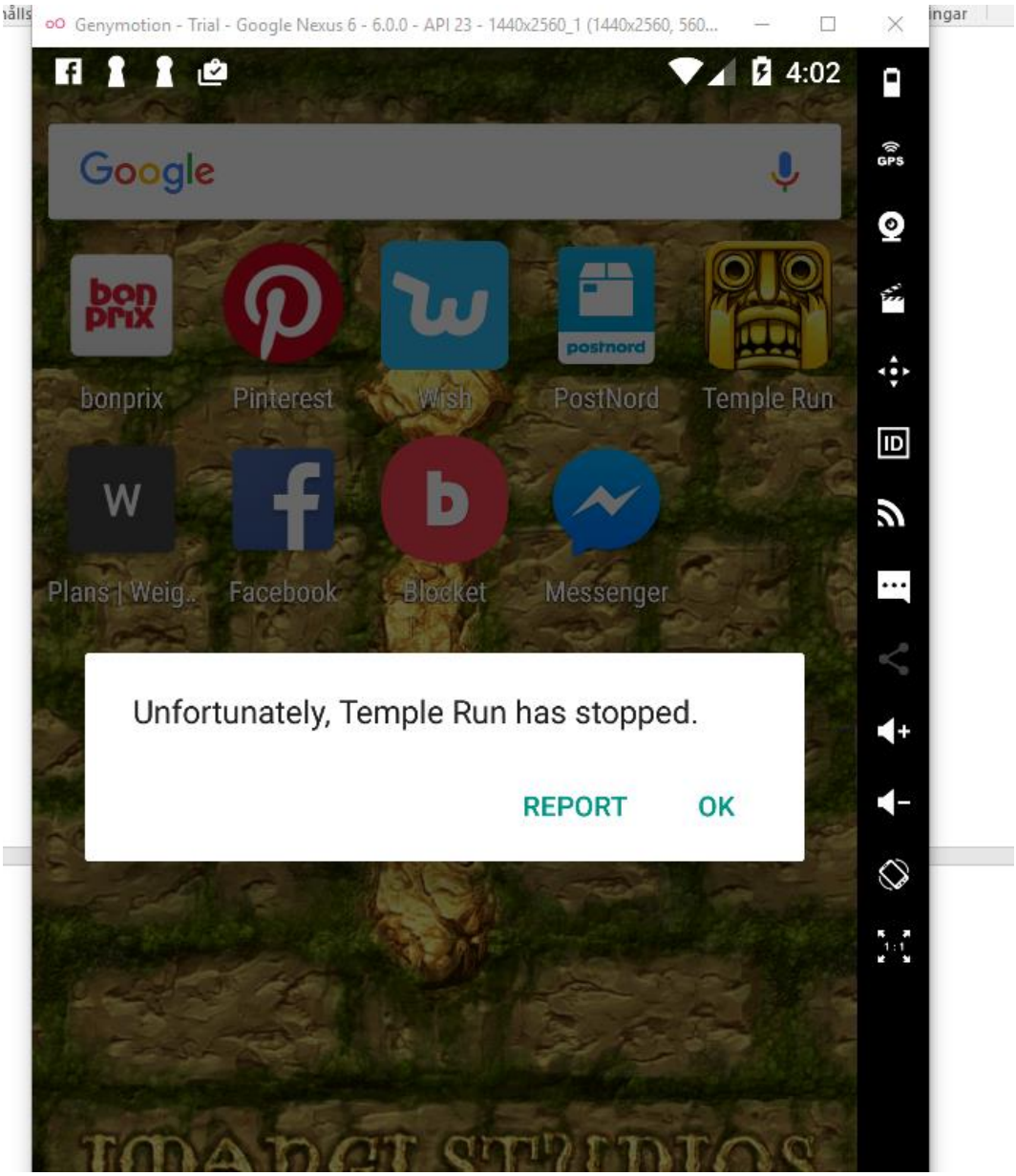

Figure 36 D3 test Application crash when intent is sent (Templerun)

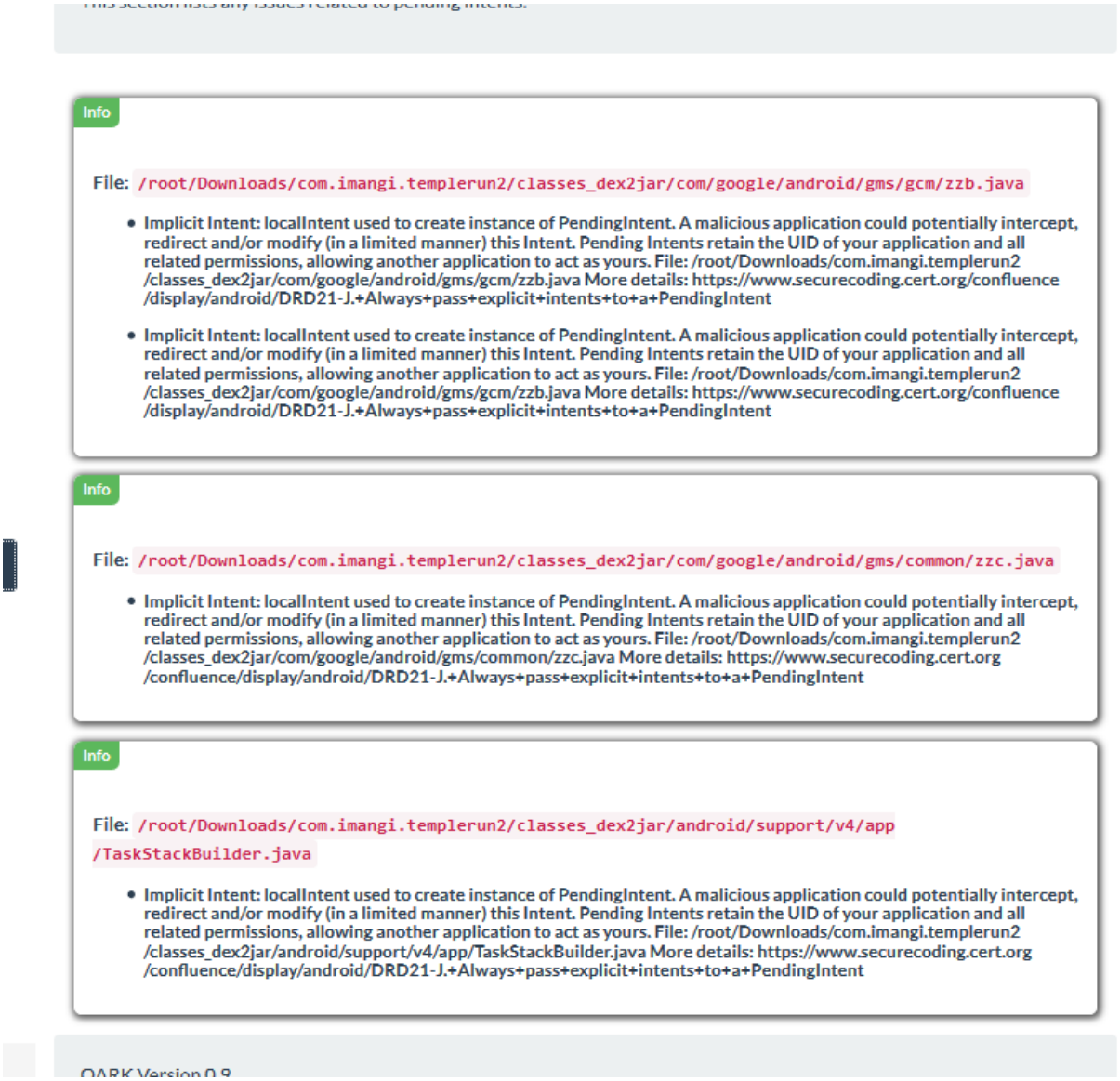

Figure 37 local pending intents (Templerun)