Masarykova univerzita
Fakulta informatiky

Analýza inštalačných APK súborov pre OS Android

Bakalárska práca

Martin Styk

Brno, jar 2016

Masarykova univerzita
Fakulta informatiky

Analýza inštalačných APK súborov pre OS Android

Bakalárska práca

Martin Styk

Brno, jar 2016

Namiesto tejto stránky vložte kópiu oficiálneho podpísaného zadania práce a prehlásenie autora školského diela.

Prehlásenie

Prehlasujem, že táto bakalárska práca je mojím pôvodným autorským dielom, ktoré som vypracoval samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používal alebo z nich čerpal, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Martin Styk

Vedúci práce: Ing. Mgr. et Mgr. Zdeněk Říha, Ph.D.

Poďakovanie

Rád by som sa poďakoval vedúcemu práce Ing. Mgr. et Mgr. Zdeňkovi Říhovi, Ph.D. za venovaný čas, ochotu a cenné pripomienky, ktoré mi pomohli pri tvorbe tejto práce.

Zhrnutie

Práca sa zaoberá získavaním metadát o inštalačných APK súboroch pre mobilný operačný systém Android. V rámci práce je vytvorená rozsiahla databáza APK balíčkov. Na základe analýzy týchto súborov sú určené štatistické vlastnosti APK súborov a príslušných aplikácií. Ako súčasť tejto práce je implementovaný nástroj na hromadné sťahovanie APK súborov, ich analýzu a výpočet štatistických dát nad množinou APK súborov. Práca sa zaoberá aj bezpečnosťou aplikácií a detekciou modifikovaných APK súborov. V práci je navrhnutá metóda detekcie upravených a prebalených APK balíčkov, ktorá je aj prakticky implementovaná. V teoretickej časti je popísaná štruktúra APK balíčkov a súborov v nich obsiahnutych.

Kľúčové slová

APK súbor, Android, Apktool, malvér, analýza aplikácií, AndroidManifest.xml

Obsah

Zoznam tabuliek

Zoznam obrázkov

1 Úvod

TBD

2 Operačný systém Android

Android je mobilný operačný systém navrhnutý primárne pre zariadenia s dotykovou obrazovkou. Android je dominantným operačným systémom na mobilných zariadeniach ako sú chytré telefóny a tablety. V treťom kvartáli roku 2015 dosahoval až 84,7% podiel na trhu operačných systémov pre mobilné zariadenia. Systém je založený na linuxovom jadre.

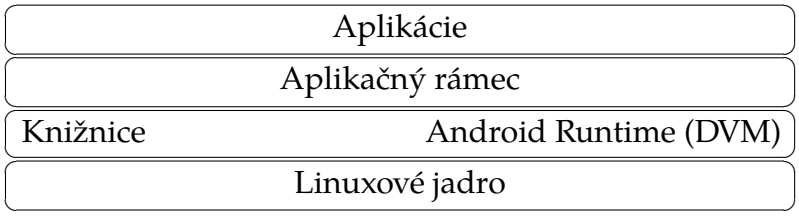
2.1 História

Začiatok vývoja operačného systému dnes známeho ako Android siaha do roku 2003, kedy vznikla spoločnosť Android, Inc.. Zakladatelia spoločnosti plánovali vývoj chytrého mobilného zariadenia, ktoré dokáže efektívne využívať prostredie a preferencie užívateľov. Prvotným zámerom bol vývoj systému pre digitálne fotoaparáty, avšak kvôli malému trhu nakoniec zvolili systém pre mobilné zariadenia. Spoločnosť Android, Inc., bola v roku 2005 kúpená spoločnosťou Google za viac ako 50 miliónov dolárov. 5. novembra 2007 bolo predstavené konzorcium Open Handset Alliance, skladajúce sa z výrobcov mobilných zariadení, mobilných operátorov a výrobcov komponentov pre mobilné zariadenia. Hlavným cieľom konzorcia je vývoj otvorených mobilných štandardov. Prvým predstaveným produktom tejto skupiny bol operačný systém založený na Linuxovom jadre – Android. Prvým komerčne dostupným chytrým telefónom s operačným systémom Android sa 22. novembra 2008 stal HTC Dream. Od roku 2008 bolo sa systém inkrementálne vylepšuje a vyvíja. Bolo vydaných množstvo opráv, vylepšení a nových funkcií. Začínajúc od verzie Android 1.5 Cupcake, je každá verzia pomenovaná podľa cukroviniek. Android je aktuálne vyvíjaný spoločnosťou Google ako open source projekt. Existuje aktívna komunita vývojárov podieľajúca sa na vývoji open source projektu Android Open Source Project. Väčšina Android zariadení sa predáva s kombináciou open source a proprietárneho softvéru. Medzi proprietárne časti zdrojového kódu patria nadstavby výrobcov telefónov a služby spoločnosti Google (Google services). Android nemá žiadny centralizovaný systém aktualizácií. To má za následok, že veľká časť zariadení často nedostáva aktualizácie. Výskum

v roku 2015 ukázal, že až 90 % zariadení obsahuje známe bezpečnostné zraniteľnosti, ktoré nie sú opravené kvôli slabej podpore.

2.2 Architektúra systému

Operačný systém Android je možné dekomponovať do piatich sekcií a štyroch základných softvérových vrstiev organizovaných v zásobníkovej štruktúre.



Obr. 2.1: Vrstevnatá architektúra systému Android

2.2.1 Linuxové jadro

Najnižšiu vrstvu predstavuje Linuxové jadro vo verzií 2.6. Jadro je upravené za účelom optimalizácie spotreby energie a operačnej pamäte, podporuje preemptívny multitasking . Táto vrstva poskytuje abstrakciu medzi hardvérom zariadenia a vyššími softvérovými vrstvami a obsahuje ovládače hardvérových komponent ako fotoaparát, dotyková obrazovka alebo sieťové rozhranie.

2.2.2 Android Runtime a Dalvik Virtual Machine

Dalvik Virtual Machine je virtuálny stroj slúžiaci na exekúciu Android aplikácií. Je obdobou Java virtuálneho stroja. Virtuálny stroj Dalvik využíva nízkoúrovňovú funkcionalitu linuxového jadra. Každá aplikácia je spustená vo vlastnom procese a na vlastnej inštancii virtuálneho stroja. Tento prístup zaručuje, že aplikácie sa navzájom neúmyselne neovplyvňujú, nepristupujú priamo k hardvéru zariadenia a využívajú abstrakciu, ktorá zabezpečuje ich platformovú nezávislosť. Od verzie Android 5.0 je virtuálny stroj Dalvik plne nahradený novým prostredím Android Runtime (ART).

2.2.3 Knižnice

Android obsahuje množstvo knižníc využívaných vývojármi alebo samotným systémom. Špecifickou skupinou sú natívne knižnice jadra, často označované ako Dalvik knižnice, ktoré obsahujú kód pre interakciu s inštanciou virtuálneho stroja ale aj napríklad knižnice pre prístup k systému súborov. Veľká časť knižníc obsiahnutá v tejto vrstve využíva natívny kód v jazyku C/C++ a slúži ako obal okolo natívneho C/C++ kódu využívajúci jazyk Java. Táto vrstva obsahuje niektoré štandardné knižnice známe z jazyka Java upravené pre využitie na operačnom systéme Android, ale aj knižnice špecifické pre platformu Android, tzv. Android knižnice.

2.2.4 Aplikačný rámec

Vrstva aplikačného rámca poskytuje vysoko-úrovňové služby používané na manažment aplikácie. Využíva koncept Android aplikácií, ktoré sa skladajú z viacerých komponent. Kľúčové služby poskytované aplikačným rámcom sú :

- Activity Manager – ovláda životný cyklus aktivít a spravuje zásobník naposledy spustených aktivít
- Content Provider – umožňuje zdieľanie dát medzi aplikáciami
- Resource Manager – poskytuje prístup k zdrojovým súborom ako reťazce, obrázky, dizajny obrazoviek
- Notification Manager – umožňuje aplikácií zobrazovať upozornenia
- View system – poskytuje prvky, ktoré tvoria grafické používateľské rozhranie aplikácie
- Package Manager – umožňuje aplikáciám zistiť informácie o ostatných aplikáciách nainštalovaných na zariadení
- Telephony Manager – umožňuje aplikáciám zistiť informácie informácie o stave telefónnych služieb
- Location Manager – poskytuje aplikácií informácie o polohe zariadenia

2.2.5 Aplikácie

Na vrchole vrstevnatej architektúry systému Android sú aplikácie, ktoré využívajú súčinnosť všetkých spomenutých vrstiev.

2.3 Aplikácie

2.3.1 Distribúcia aplikácií

Apk súbory predstavujúce inštalačné balíčky Android aplikácií sú najčastejšie distribuované pomocou obchodu s aplikáciami. Oficiálny obchod pre Android zariadenia je Google Play [1]. Aplikácie môžu byť distribuované pomocou alternatívnych obchodov ako napríklad Amazon Appstore[2] alebo SlideMe [3]. Operačný systém Android v základom nastavení neumožňuje inštaláciu aplikácií z iných zdrojov ako Google Play. Inštalácia z neznámych zdrojov môže byť povolená v nastaveniach zariadenia. Inštalačné APK balíčky je možné získať aj zo stránok na zdieľanie ľubovoľného obsahu, na ktorých sa často distribuujú aplikácie ktoré sú v oficiálnych zdrojoch platené. Takéto súbory sú často modifikované a môžu obsahovať potenciálny škodlivý kód. Často ich označujeme ako prebalené[4] aplikácie. Viacej o takýchto aplikáciách sa dozviete v kapitole TODO.

2.3.2 Inštalácia aplikácií

Aplikácie môžeme rozdeliť na dve skupiny:

- Predinštalované aplikácie – často označované aj ako systémové aplikácie. Tieto aplikácie sú nainštalované spolu so systémom a často nemôžu byť bežným používateľom odinštalované. Príkladom je základná aplikácia pre fotoaparát, kontakty alebo telefón
- Aplikácie nainštalované používateľom – Aplikácie nainštalované jedným z nasledujúcich spôsobov

1. https://play.google.com/store
2. http://www.amazon.com/mobile-apps/b?node=2350149011
3. http://slideme.org/
4. angl. repackaged

- prostredníctvom obchodu s aplikáciami, najčastejšie Google Play Store
- prostredníctvom nástroja Android Debug Bridge (ADB) ktorý je obsiahnutý v Android SDK a umožňuje inštaláciu a ladenie aplikácií na zariadení pripojenom k počítaču pomocou USB kábla
- otvorením apk balíčka umiestneného v zariadení

Základnou aplikáciou starajúcou sa o inštaláciu apk balíčkov je *PackageInstaller*, ktorý poskytuje užívateľské rozhranie na komunikáciu so službou *PackageManager*. *PackageManager* poskytuje v rámci triedy *PackageManagerService.java* API pre inštaláciu, aktualizáciu a odinštaláciu aplikácií. Natívny démon installd prijíma požiadavky od služby *PackageManagerService.java* s ktorou komunikuje prostredníctvom lokálneho soketu */dev/socket/installed*. Služba *PackageManager* a démon *installd* sú spustené pri štarte systému. *PackageManager* čaká na pridanie požiadavky na inštaláciu do zoznamu inštalovaných aplikácií. Pri inštalácií analyzuje súbor *AndroidManifest.xml* a relevantné informácie ukladá do súborov */data/system/packages.xml* a */data/system/packages.list*. Priečinok, do ktorého sa apk súbor rozbalí, vytvára démon *installd*, o rozbalenie a kopírovanie obsahu sa stará *PackageManager*. Predinštalované aplikácie sú inštalované do zložky */system/app/*, aplikácie inštalované užívateľom do zložky */data/app/*. Súbor classes.dex, ktorý je obsiahnutý v APK balíčku je kopírovaný do */data/dalvik-cache/*. *PackageManager* vytvorí priečinok */data/data/nazov_balíčku* v ktorom sa nachádzajú preferencie, databázy alebo natívne knižnice aplikácie.

3 APK súbory

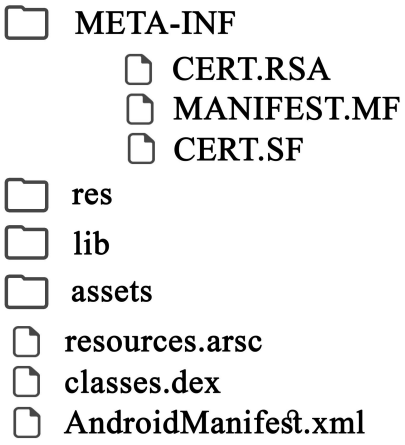
APK súbory sú balíčky používané operačným systémom Android. Celý názov ukrytý za skratkou APK je Android application package file. Tieto súbory slúžia na distribúciu aplikácií v operačnom systéme Android. Ich použitie a význam je analogický ako pri MSI[1] balíčkoch používaných v systéme Microsoft Windows, alebo DEB[2] balíčkoch používaných v niektorých linuxových distribúciách. APK súbory sú asociované s príponou .apk a príslušný MIME typom *application/vnd.android.package-archive*.

Štruktúra APK balíčkov vychádza z JAR[3] balíčkov – súborov používaných na distribúciu aplikácií alebo knižníc na platforme Java. Formát APK rozširuje všeobecnejší JAR formát o súbory, ktoré sú špecifické pre cieľovú platformu, ktorou je operačný systém Android. Zároveň si však ponecháva vlastnosti JAR súborov. APK balíčky sú archívne súbory v ZIP[4] formáte. Keďže APK používajú ZIP formát, k ich obsahu môžeme jednoducho pristúpiť rozbalením archívu štandardným spôsobom. APK súbory vznikajú ako výstup kompletnej kompilácie a zabalenia aplikácií pre Android. APK súbor každej aplikácie obsahuje všetky potrebné súbory na jej inštaláciu a spustenie. Medzi týmito súbormi sa typicky nachádza *classes.dex* súbor obsahujúci skompilovaný zdrojový kód, *resources.arsc* súbor ktorý obsahuje skompilované zdroje aplikácie, súbor *AndroidManifest.xml* a neskompilované súbory ako sú napríklad obrázky.

3.1 Priečinok META-INF

Priečinok obsahujúci súbory, ktorých úlohou je zaručiť integritu ostatných súborov v APK balíčku a s ňou spojenú bezpečnosť celého

1. `WindowsInstallerPackagehttps://technet.microsoft.com/en-us/library/cc978328.aspx`
2. `https://www.debian.org/doc/manuals/debian-faq/ch-pkg_basics.en.html`
3. Java archive
4. `https://en.wikipedia.org/wiki/Zip_(file_format)`



Obr. 3.1: Typická štruktúra APK súboru

systému. V prípade detekcie pozmenených súborov a narušenia integrity operačný systém Android nedovolí inštaláciu APK balíčku. Po každej zmene je nutné balíček digitálne podpísať.

CERT.RSA

Súbor obsahujúci verejný kľúč ktorý slúži na overenie digitálneho podpisu balíčka.

MANIFEST.MF

Súbor obsahujúci relatívne cesty a SHA-1 hashe [5] všetkých súborov v APK balíčku. Tento súbor neobsahujú len APK súbory, je typický pre každý JAR archív.

Typický začiatok súboru *MANIFEST.MF* vyzerá nasledovne:

```
Manifest-Version: 1.0
Built-By: 0.12.2
Created-By: Android Gradle 0.12.2
```

5. reťazce v Base64 kódovaní

```
Name: res/drawable-xhdpi-v4/libraries.png
SHA1-Digest: VvgaO1jpW3iS1nBBikD/urdbN58=

Name: res/layout/activity_settings.xml
SHA1-Digest: 1coP1lt9Lmccc7SMZGHxNv4bbKs=
```

CERT.SF

Súbor podobný ako *MANIFEST.MF*, avšak namiesto SHA-1 hashov samotných súborov obsahuje SHA-1 hashe záznamov o týchto súboroch z *MANIFEST.MF*. Okrem toho obsahuje aj hash celého súboru *MANIFEST.MF*.
Záznam o jednom súbore v APK balíčku v súbore *CERT.SF* vyzerá nasledovne:

```
Name: res/drawable-xhdpi-v4/libraries.png
SHA1-Digest: Slg56lqothjvmaBikD/urdb7q6=
```

Reťazec *Slg56lqothjvmaBikD/urdb7q6=* reprezentuje SHA-1 hash nasledujúceho záznamu zo súboru *MANIFEST.MF*:

```
"Name: res/drawable-xhdpi-v4/libraries.png
SHA1-Digest: VvgaO1jpW3iS1nBBikD/urdbN58=
"
```

3.2 Priečinok res

Priečinok obsahujúci zdrojové súbory ako napríklad obrázky, zvuky alebo ikony. Okrem multimediálnych súborov obsahuje taktiež zdrojové XML súbory určujúce vzhľad obrazoviek, použité grafické štýly, alebo texty použité v aplikácii. Niektoré z týchto XML súborov môžu byť skompilované do binárneho formátu. V zdrojovom kóde sú tieto zdroje odkazované pomocou unikátnych identifikátorov. Identifikátory sú generované počas kompilácie nástrojom aapt a nachádzajú sa v projektovej triede *R*. Pre každý typ zdrojového súboru je generovaná podtrieda triedy *R*. Všetky zdroje aplikácie by mali byť externalizo-

vané a uložené v špecifickom podpriečinku v tomto adresári.

Podporované podpriečinky

- animator - obsahuje XML súbory definujúce property animácie[**article-full**]
- anim - obsahuje XML súbory definujúce tween animácie, môže obsahovať aj property animácie
- color - obsahuje XML súbory definujúce farby a ich zmeny na základe stavu objektov na ktoré sú aplikované
- drawable - obsahuje obrázky vo formáte PNG, 9.PNG, JPG, GIF alebo XML súbory skompilované do formy vykresliteľných obrázkov
- mipmap - ikony aplikácie s rôznou hustotou pixelov
- layout - obsahuje súbory vo formáte XML definujúce vzhľad rozmiestnenie prvkov na obrazovke
- menu - obsahuje súbory XML definujúce menu aplikácie
- raw - obsahuje súbory ktoré musia byť uložené a použité v neskomprimovanej forme a kvalite
- values - obsahuje súbory vo formáte XML definujúce hodnoty textových reťazcov, farieb, štýlov, základných rozmerov
- xml - obsahuje XML súbory, ktoré môžu byť načítané počas behu aplikácie

V spomenutých priečinkoch sú uložené základné zdroje aplikácie. Tieto zdrojové súbory určujú základný dizajn a obsah aplikácie. Avšak rôzne typy Android zariadení môžu využívať rôzne zdrojové súbory. Alternatívne zdroje sa využívajú na prispôsobenie dizajnu a obsahu veľkosti a aktuálnej konfigurácií zariadenia. Sú umiestnené v priečinku, ktorého názov pozostáva z typu zdrojového súboru, ktorý korešponduje so základným názvom priečinku a názvu hodnoty konfiguračného atribútu pre ktorý je tento priečinok určený. Je možné kombinovať viacero konfiguračných atribútov.

Najpoužívanejšie atribúty

- Jazyk a región – jazyk je definovaný podľa ISO 639-1[6] kódovania s možnosťou rozšírenia pomocou ISO 3166-1-alpha-2 regionálneho kódu. Napríklad obrázky špecifické pre zariadenia s francúzskym jazykom sa nachádzajú v priečinku *res/drawable-fr*

- Veľkosť obrazovky – v závislosti na veľkosti a rozlíšení obrazovky rozlišuje štyri možné hodnoty – small, normal, large, xlarge

- Orientácia obrazovky – v závislosti na orientácii zariadenia hodnota port pre zariadenie vo vertikálnej polohe, hodnota land pre polohu horizontálnu

- Hustota obrázkových bodov obrazovky - hodnoty určujúce vhodnosť zdrojových súborov vzhľadom na hustotu obrázkových bodov obrazovky daného zariadenia

3.3 Priečinok lib

Priečinok obsahujúci skompilovaný zdrojový kód natívnych knižníc. Tieto knižnice sú špecifické pre typ procesora. V závislosti na architektúre procesora obsahuje podpriečinky : *armeabi, armeabi-v7a, arm64-v8a, x86, x86_64, mips*.

3.4 Priečinok assets

Priečinok obsahujúci súbory uložené a používané v originálnej neskomprimovanej forme. V tomto priečinku sa často nachádzajú textové súbory, html súbory, licenčné informácie, obrázky alebo textúry. Na rozdiel od priečinku *res/raw/*, zdrojovým súborom v umiestneným v priečinku assets nie sú pridelené unikátne identifikátory uložené v triede *R.java*. K súborom sa pristupuje ako k dátam uloženým v bežnom súborovom systéme. Trieda *AssetManager* poskytuje funkcionalitu na čítanie súborov ako prúdu bytov a navigovanie v tomto priečinku.

6. http://www.iso.org/iso/home/standards/language_codes.htm

3.5 resources.arsc

Súbor obsahujúci mapovanie medzi zdrojovými súbormi z priečinku *res* a ich identifikátormi. Vytvára sa počas kompilácie. Obsahuje XML súbory v binárnom formáte.

3.6 classes.dex

Classes.dex je súbor obsahujúci skompilovaný zdrojový kód aplikácie. Zdrojové súbory Android aplikácií sú napísané v jazyku Java. Java súbory sú skompilované do Java bytekódu pomocou bežného kompilátoru pre platformu Java. Výsledkom tejto kompilácie sú súbory s príponou *class*, ktoré sú následne preložené do Dalvik bytekódu pomocou nástroja *dx* ktorý je súčasťou Android Software Developement Kit. Výstupom nástroju *dx* je jediný súbor obsahujúci skompilovaný celý výkonný zdrojový kód aplikácie – *classes.dex*. Tento súbor je skomprimovanou a optimalizovanou verziou všetkých *class* súborov. Takto skompilovaný program môže byť vykonaný len vo virtuálnom stroji Dalvik, alebo v novšom prostredí ART (Android Runtime) používanom primárne od verzie Android 5.0 „Lollipop“.

3.7 AndroidManifest.xml

http://developer.android.com/guide/topics/manifest/manifest-intro.html
Súbor ktorý musí obsahovať každá Android aplikácia. Tento súbor poskytuje informácie o aplikácii operačnému systému Android. Neobsahuje žiadny výkonný kód. Definuje meno a verziu, ktoré slúžia ako unikátny identifikátor danej aplikácie. Popisuje všetky komponenty z ktorých sa aplikácia skladá, cesty k použitým knižniciam, minimálny vyžadovaný level Android API, oprávnenie vyžadované aplikáciu na prístup k chráneným častiam Android API a taktiež oprávnenia, ktoré sú vyžadované od iných komponent pri pokuse komunikovať s danou aplikáciou. Súbor *AndroidManifest.xml*, ktorý nájdeme v APK balíčku je vo formáte binárneho XML súboru. Je ho však možné previesť do klasického čitateľného XML formátu.

Keďže *AndroidManifest.xml* je základným súborom poskytujúcim metadáta o Android aplikácii a APK súbore, informácie získané z

tohto súboru tvoria veľkú časť štatistických dát zbieraných a vyhodnocovaných v tejto práci. Preto sa detailnejšie pozrieme na jeho štruktúru a niektoré dôležité elementy, ktoré obsahuje.
Nasledujúci diagram ukazuje základnú štruktúru tohto XML súboru.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest>
    <uses-permission />
    <permission />
    <uses-sdk />
    <uses-configuration />
    <uses-feature />
    <supports-screens />
    <compatible-screens />
    <application>
        <activity />
        <service/>
        <receiver/>
        <provider/>
        <uses-library />
    </application>
</manifest>
```

Kód 3.1: Základná štruktúra súboru AndroidManifest.xml

3.7.1 Element manifest

```
<manifest xmlns:android="URL"
          package="string"
          android:sharedUserId="string"
          android:sharedUserLabel="string resource"
          android:versionCode="integer"
          android:versionName="string"
          android:installLocation=["auto" | "internalOnly" |
                                   "preferExternal"] >
</manifest>
```

AndroidManifest.xml obsahuje element manifest ako koreňový prvok. Tento element je povinný a každý manifest ho obsahuje práve jeden.

Element manifest definuje atribúty:

- xmlns:android – povinný atribút definujúci menný priestor
- package – meno balíku aplikácie, povinný atribút definujúci identitu aplikácie. Meno balúku nemôže byť zmenené
- android:sharedUserId – identifikátor aplikácie zdieľaný s ostatnými aplikáciami za účelom vzájomnej komunikácie
- android:sharedUserLabel – čitateľná podoba android:sharedUserId identifikátoru
- android:versionCode – interná informácia o verzii aplikácie. Tento atribút je využívaný len na rozoznanie novších verzií od starších. Novšie aplikácie obsahujú vyššiu hodnotu
- android:versionName – informácia o verzií aplikácie prezentovaná užívateľom. Pre systém Android neposkytuje informáciu o verzií aplikácie, tú obsahuje atribút android:versionCode.
- android:installLocation – určuje miesto základné miesto inštalácie aplikácie. Pokiaľ je aplikáciu možné nainštalovať len do vnútornej pamäti zariadenia, obsahuje hodnotu internalOnly. Takáto aplikácia nemôže byť presunutá na externé pamäťové médium (typicky SD karta). Táto hodnota je základnou použitou možnosťou, ak tento atribút nie je definovaný. V prípade hodnoty auto sú aplikácie inštalované vo vnútornej pamäti, ale môžu byť presunuté do pamäti externej. Hodnota preferExternal zabezpečí, že systém sa pokúsi o inštaláciu na externé pamäťové médium. V prípade neúspechu sa použije interná pamäť

3.7.2 Element uses-permission

```
<uses-permission android:name="string"
       android:maxSdkVersion="integer" />
```

Prístup k niektorým dátam alebo častiam kódu je z dôvodu ochrany limitovaný. Android využíva princíp povolení[7]. Povolenia môžu byť definované samotnou aplikáciou, inou aplikáciou alebo systémom Android. Aplikácia, ktorá chce pristupovať k chráneným dátam alebo používať chránené časti kódu, musí pomocou tagu uses-permissions deklarovať vyžadované povolenia. Prístupové povolenia sú aplikácii schválené užívateľom. Vo verzii Android 5.1 a starších, systém počas inštalácie oboznámi užívateľa so všetkými povoleniami, ktoré aplikácia vyžaduje. V prípade, že ich používateľ neschváli, aplikácia nebude nainštalovaná. Od verzie Android 6.0 užívateľ schvaľuje povolenia počas behu aplikácie.

Element uses-permission definuje atribúty:

- android:name – definuje názov povolenia
- android:maxSdkVersion – najvyšší level Android API, pre ktorý je dané povolenie potrebné

3.7.3 Element permission

```
<permission android:description="string resource"
            android:icon="drawable resource"
            android:label="string resource"
            android:name="string"
            android:permissionGroup="string"
            android:protectionLevel=["normal"|"dangerous"|
                                     "signature"|
                                     "signatureOrSystem"] />
```

Definuje bezpečnostné povolenie, ktoré môže byť použité na obmedzenie prístupu ku komponente aplikácie. Toto povolenie je následne používané aplikáciami, ktoré vyžadujú prístup k chránenej časti danej aplikácie.

Najdôležitejšie atribúty definované v rámci elementu permission:

7. angl. permissions

- android:name – názov povolenia, aplikácie vyžadujúce dané povolenie uvádzajú túto hodnotu v atribúte android:name v tagu uses-permission. Názov musí byť unikátny a mal by dodržiavať konvencie pomenovávania jazyka Java.

- android:permissionGroup – priradí toto povolenie do skupiny povolení (angl. Permission group). Táto skupina povolení musí byť deklarovaná v rámci elementu permission-group v niektorej z nainštalovaných aplikácií. Slúži na logické zoskupenie významovo podobných oprávnení.

- android:protectionLevel – charakterizuje potenciálne riziko spojené s použitím daného povolenia. Táto hodnota určuje postup operačného systému pri rozhodovaní o udelení povolenia. Základnou hodnotou je „normal“, ktorá reprezentuje povolenia s nízkou mierou rizika. Tieto povolenia sú aplikácii automaticky schválené systémom Android počas inštalácie. Pre povolenia z zvýšenou mierou rizika je určená hodnota „dangerous“. Tieto povolenia typicky udeľujú aplikácií prístup k citlivým dátam alebo k prvkom Android zariadenia, ktoré môžu negatívne ovplyvniť jeho používanie. Pretože tento typ povolení prináša potenciálne riziko, systém ho nemôže udeliť automaticky, ale až po explicitnom súhlase používateľa. V prípade, že tento atribút obsahuje hodnotu „signature“, povolenie je udelené automaticky a to len aplikáciám, ktorých certifikát je rovnaký ako certifikát aplikácie definujúcej dané povolenie. Hodnota „signatureOrSystem“ rozširuje hodnotu „signature“ a povolenia sú udelené aj aplikáciám Android system image.

3.7.4 Element uses-sdk

```
<uses-sdk android:minSdkVersion="integer"
          android:targetSdkVersion="integer"
          android:maxSdkVersion="integer" />
```

Element vyjadrujúci kompatibilitu aplikácie s verziou Androidu pomocou čísla verzie Android API.

Atribúty definované elementom uses-sdk:

- android:minSdkVersion – vyjadruje najnižší level API vyžadovaný aplikáciou. Pokiaľ je táto hodnota vyššia ako API úroveň zariadenia, systém zabráni inštalácií. Tento atribút by mala obsahovať každá aplikácia
- android:targetSdkVersion – informuje o úrovni API na ktorej bola aplikácia testovaná a pre ktorú je primárne určená. V prípade, že zariadenie podporuje vyššiu verziu Android API ako definuje spomínaný atribút, aplikácia môže byť spustená v móde kompatibility s touto verziou API
- android:maxSdkVersion – najvyššia úroveň API kompatibilná s aplikáciou. Z dôvodu spätnej kompatibility nie je používanie tohto atribútu doporučené. Tento atribút bol využívaný len do verzie Android 2.0.1

3.7.5 Element uses-feature

```
<uses-feature
  android:name="string"
  android:required=["true" | "false"] />
```

Deklaruje hardvérovú alebo softvérovú vlastnosť [8] používanú aplikáciou. Tento element informuje externé entity o funkciách, ktoré aplikácia vyžaduje. Deklarované vlastnosti majú informačný charakter. Systém Android nekontroluje či zariadenie podporuje všetky vlastnosti deklarované aplikáciou. Tento element je využívaný službou Google Play na filtrovanie aplikácií vyhovujúcich danému zariadeniu, a preto by mala byť deklarovaná každá vyžadovaná vlastnosť.

Element obsahuje nasledujúce atribúty:

- android:name – názov vyžadovanej vlastnosti
- android:required – určuje, či aplikácia vyžaduje danú vlastnosť pre korektné fungovanie. Pokiaľ obsahuje hodnotu true,

8. angl. feature

aplikácia nie je schopná korektne fungovať na zariadení nepodporujúcom danú vlastnosť

http://developer.android.com/guide/topics/manifest/uses-feature-element.html

3.7.6 Element supports-screens

```
<supports-screens android:resizeable=["true"| "false"]
                  android:smallScreens=["true" | "false"]
                  android:normalScreens=["true" | "false"]
                  android:largeScreens=["true" | "false"]
                  android:xlargeScreens=["true" | "false"]
                  android:anyDensity=["true" | "false"]
                  android:requiresSmallestWidthDp="integer"
                  android:compatibleWidthLimitDp="integer"
                  android:largestWidthLimitDp="integer"/>
```

Špecifikuje podporované typy obrazoviek. V prípade inštalácie na zariadení s väčšou obrazovkou ako aplikácia podporuje, informuje systém o potrebe využitia módu obrazovej kompatibility.

Väčšina atribútov deklaruje podporované veľkosti obrazoviek. Ďalšie atribúty využívané v tejto práci sú:

- android-resizeable – indikuje schopnosť aplikácie korektne sa prispôsobiť rôznym veľkostiam obrazoviek
- android:anyDensity – indikuje či aplikácia obsahuje zdrojové súbory vhodné pre obrazovky s rôznou hustotu obrazových bodov

3.7.7 Element activity

Deklaruje aktivity. Aktivity sú základnými časťami aplikácie. Sú to triedy rozširujúce triedu *android.app.Activity*. Sú zamerané na jednotlivé prípady použitia aplikácie, implementujú časť grafické užívateľského rozhrania a zabezpečujú komunikáciu s užívateľom. Tieto triedy sú časťou zdrojového kódu a v skompilovanej forme sa nachádzajú v

súbore classes.dex. Pri viacerých spustených aktivitách zohráva dôležitú úlohu životný cyklus aktivity. Viac informácií o životnom cykle aktivít nájdete na (http://developer.android.com/reference/android/app/Activity.html). Všetky aktivity musia byť deklarované pomocou tagu activity. V prípade, že deklarované nie sú, systém ich bude ignorovať a nebudú spustené. Syntax a kompletný zoznam atribútov definovaných v tagu activity môžete nájsť na http://developer.android.com/guide/topics/manifest/activity-element.html.

3.7.8 Element service

Deklaruje komponenty aplikácie typu služba[9]. Na rozdiel od aktivít, služby nemajú grafické používateľské rozhranie. Slúžia na implementáciu dlhodobých úloh na pozadí, ktoré môžu bežať aj v čase keď aplikácia nie je aktívna na popredí, alebo poskytujú funkcionalitu využívanú aplikáciou. Rozlišujeme dva typy služieb. Služby typu started po spustení bežia pokým nedokončia svoju úlohu a ostatným komponentom nevracajú výsledok. Služby typu bound komunikujú s ostatnými komponentami pomocou modelu klient-server a sú ukončené keď pre nich neexistuje klient. Každá služba rozširuje základnú triedu *android.app.Service* a musí byť deklarovaná pomocou elementu service, inak bude ignorovaná. Syntax prvku service môžete nájsť na http://developer.android.com/guide/topics/manifest/service-element.html

3.7.9 Element provider

Deklaruje komponenty aplikácie ktorých úlohou je poskytovanie štrukturovaného prístupu k dátam spravovaným aplikáciou. Poskytovatelia obsahu [10] sú implementovaný ako podtriedy triedy *android.content.ContentProvider*. Využitie tejto komponenty je nutné len v prípade potreby zdieľania dát medzi viacerými aplikáciami. Operačný systém Android si ukladá referencie na jednotlivých poskytovateľov obsahu pomocou authority textového reťazca, ktorý je definovaný ako jeden z atribútov elementu provider. Kompletný list atribútov sa nachádza na http://developer.android.com/guide/topics/manifest/provider-element.html.

9. angl. service
10. angl. content providers

Systém Android si nerozpoznáva poskytovateľov obsahu nedefinovaných v AndroidManifest.xml.

3.7.10 Element receiver

Deklaruje komponentu aplikácie implementovanú ako podtriedu triedy *android.content.BroadcastReceiver*. Tieto komponenty umožňujú aplikácií prijímať a reagovať na informácie o zámere spustenia aktivity (intent) aj v čase, keď ostatné komponenty aplikácie nie sú spustené. Tieto informácie sú vysielané systémom alebo inou aplikáciou. Systém Android môžeme oboznámiť s existenciou komponent typu broadcast receiver pomocou tagu receiver alebo aj dynamicky pomocou volania *Context.registerReceiver()* v zdrojovom kóde aplikácie. Kompletný list atribútov sa nachádza na http://developer.android.com/guide/topics/manifest/receiver-element.html

3.7.11 Element uses-library

```
<uses-library
  android:name="string"
  android:required=["true" | "false"] />
```

Špecifikuje zdieľanú knižnicu vyžadovanú aplikáciou. Tento element informuje systém o potrebe zahrnúť cestu k zdrojovému kódu knižnice medzi cesty v ktorých Dalvik Virtual Machine hľadá zdrojové súbory. V angličtine sú tieto cesty označované ako class path. Najpoužívanejšie android balíky ako napríklad android.app, android.content alebo android.view sú obsiahnuté v základnej knižnici, ktorá je automaticky pripojená ku každej aplikácií a nemusia byť deklarované týmto elementom. Balíčky, ktoré nie sú v základnej android knižnici, musia byť deklarované. Tento element ovplyvňuje inštaláciu aplikácie na konkrétnom zariadení a taktiež dostupnosť aplikácie v obchode Google Play.

Definuje atribúty:

- android:name – názov knižnice, ktorý sa nachádza v dokumentácií príslušného použitého balíčku

- android:required – indikuje či aplikácia potrebuje knižnicu špecifikovanú atribútom android:name. V prípade hodnoty true aplikácia nie je schopná fungovať bez danej knižnice. Systém zamietne inštaláciu takejto aplikácie, pokiaľ zariadenie neobsahuje danú knižnicu. Ak je tento atribút nastavený na hodnotu false, aplikácia je schopná korektne fungovať aj bez danej knižnice

4 Databáza inštalačných APK súborov

Základnou úlohou tejto práce je vytvoriť dostatočne veľkú databázu inštalačných APK balíčkov. Pre ďalšie potreby práce bolo požadované, aby veľká časť aplikácií pochádzala z neoficiálnych zdrojov, čím sa zvyšuje pravdepodobnosť, že aplikácia obsahuje malvér.
Naša databáza pozostáva približne z 20000 Android aplikácií. Tie boli zaobstarané v časovom rozmedzí medzi novembrom 2015 a februárom 2016. Žiadna z aplikácií nebola stiahnutá priamo z obchodu Google Play, ale veľká časť bola získaná pomocou projektu Playdrone vyvíjaného na University of Columbia (http://www.cs.columbia.edu/ nieh/pubs/sigmetrics2014_playdrone.pdf). V rámci tohto projektu bolo v novembri 2014 z Google Play stiahnutých viac ako milión aplikácií dostupný pre zariadenie Galaxy Nexus s operátorom T-Mobile. Naša databáza obsahuje 8200 najsťahovanejších aplikácií z Google Play v období november 2014, ktoré boli stiahnuté z archívu projektu Playdrone. Celková veľkosť všetkých stiahnutých APK súborov je 192 GB. Prehľad všetkých zdrojov APK súborov a ich počet zobrazuje tabuľka 4.1

Zdroj	Počet stiahnutých aplikácií
Playdrone[1]	8200
www.appsapk.com	6470
www.apkmaniafull.com	2870
www.androidapksfree.com	1030
www.zippyshare.com	750
torrenty	550
www.uloz.to	190
Spolu	20060

Tabuľka 4.1: Zdroje prevzatých APK súborov

4.1 Implementácia

Viac ako 90 % aplikácií bolo stiahnutých automatizovane prostredníctvom aplikácie *ApkDownloader* implementovanej v rámci tejto práce. Aplikácia je implementovaná v jazyku Java a využíva nástroj Maven na automatizovanie build procesu. Neposkytuje grafické užívateľské rozhranie, ale užívateľ môže zadávať parametre prostredníctvom príkazového riadku. Podporuje sťahovanie aplikácií získaných pomocou projektu Playdrone alebo z lokalít www.appsapk.com, www.apkmaniafull.com, www.androidapksfree.com. Aplikácia funguje na jednoduchom princípe, keď najskôr získa zoznam URL odkazov na APK súbory, ktoré následne stiahne. Užívateľ pomocou parametrov špecifikuje z ktorej podporovanej lokality chce APK súbory stiahnuť, ich želaný počet, umiestnenie prebraných súborov a maximálny počet súbežných preberaní. Pri vyhľadávaní URL odkazov je na prácu s HTML súbormi použitá open source knižnica *jsoup*. Pri sťahovaní sa využíva knižnica *HtmlUnit*, ktorá poskytuje funkcionalitu internetového prehliadača. Na preberanie súborov z URL odkazov je taktiež použitá knižnica *Apache Commons IO*. Keďže je *ApkDownloader* open source, môže byť jednoducho rozšírený o podporu sťahovania APK súborov z nových lokalít. Torrent súbory boli získane automatizovane s využitím knižnice *flux*[2].

2. https://github.com/ProjectMoon/flux

5 Analýza APK súborov

https://www.rsaconference.com/writable/presentations/file_upload/stu-w02b-beginners-guide-to-reverse-engineering-android-apps.pdf http://www.sans.org/reading-room/whitepapers/pda/reverse-engineering-malware-android-33769 Hlavnou úlohou práce je získať informácie o APK súboroch ich detailnou analýzou. APK súbory majú pevnú štruktúru a jednoduchý formát, vďaka čomu je možná ich analýza a reverzné inžinierstvo. Reverzné inžinierstvo je proces analýzy funkcionality a obsahu aplikácie. Keďže APK súbory využívajú ZIP formát, mnohé informácie je možné získať jednoduchým rozbalením. Základnou úlohou analýzy je získanie metadát o APK súbore.

5.1 Nástroje reverzného inžinierstva

Existuje viacero nástrojov poskytujúcich funkcionalitu pre reverzné inžinierstvo Android aplikácií. Okrem aplikácií tretích strán je možné vo veľkej miere použiť aj štandardné nástroje obsiahnuté v Android Software Development Kit.

5.1.1 ApkTool

(http://ibotpeaches.github.io/Apktool/) Nástroj na reverzné inžinierstvo Android aplikácií. Dokáže dekódovať zdroje aplikácie do takmer originálnej podoby. Do čitateľnej podoby prevádza súbory resources.arsc, classes.dex aj binárne XML súbory. Z dekódovaných súborov umožňuje opätovné zostavenie APK súboru. Súbor classes.dex je dekompilovaný do súborov vo formáte SMALI. Smali súbory obsahujú nízko úrovňový kód na úrovni asembleru. ApkTool podporuje debugovanie smali kódu.

5.1.2 Dex2Jar

(https://github.com/pxb1988/dex2jar) http://pof.eslack.org/2011/02/18/from-apk-to-readable-java-source-code-in-3-easy-steps/ Nástroj podporujúci dekódovanie DEX súborov do formátu skompilovaných CLASS súborov http://fileinfo.com/extension/class .Výsledné CLASS sú-

bory môžu byť prevedené do čitateľného kódu v jazyku Java pomocou dekompilátoru JD-GUI(http://jd.benow.ca/) Pracuje výhradne so súborom classes.dex a nepodporuje prevod binárnych XML do čitateľnej podoby.

5.1.3 AXML

https://github.com/xgouchet/AXML AXML je knižnica navrhnutá na prácu s binárnymi XML súbormi, ktoré vznikajú počas zostavenia Android aplikácie pomocou nástroja AAPT. Knižnica umožňuje prevod takýchto XML súborov do čitateľného XML formátu, je implementovaná v jazyku Java.

5.1.4 AAPT

http://elinux.org/Android_aapt Android Asset Packaging Tool (AAPT) je štandardný nástroj obsiahnutý v Android Software Development Kit (SDK). Android SDK je kolekcia nástrojov používaných pri vývoji a zostavení Android aplikácií. Nástroj AAPT umožňuje vytvorenie, aktualizovanie a prezeranie súborov vo formáte APK. Dokáže skompilovať zdrojové súbory do binárnej formy a umožňuje aj ich dekompiláciu.

5.2 Implementácia analýzy

Analýza APK súborov je implementovaná v rámci programu ApkAnalyzer a môže byť spustená pomocou argumentu –analyze. Zároveň je potrebné špecifikovať analyzovaný APK súbor alebo priečinok obsahujúci takéto súbory pomocou argumentu –in a priečinok do ktorého bude zapísaný výstup analýzy pomocou argumentu –out. ApkAnalyzer je aplikácia prispôsobená na prácu s veľkým počtom APK súborov, proces analýzy je preto paralelizovaný a každé dostupné procesorové jadro analyzuje inú aplikáciu. Pre každú analyzovanú aplikáciu je vygenerovaný výstupný súbor vo formáte JSON obsahujúci získané metadáta o danej aplikácií.

Zbierané metadáta je možné rozdeliť do piatich kategórií:

- Základné informácie o APK súbore – v tejto kategórií sa nachádzajú informácie ako je veľkosť APK súboru alebo súborov

classes.dex a *resources.arsc*. Pre získanie veľkosti súborov obsiahnutých v APK balíčku je balíček rozbalený do dočasného adresára

- Informácie zo súboru *AndroidManifest.xml* – *AndroidManifest.xml* predstavuje hlavný zdroj meta informácií o aplikácii pre systém Android(viď 3.7). Dáta nachádzajúce sa v tomto súbore tvoria významnú časť dát získaných našou analýzou. Na prevod z binárneho XML formátu je primárne použitá knižnica AXML (viď 5.1.3), v prípade, že prevod zlyhá, použije sa nástroj ApkTool (viď 5.1.1). Dáta analýzou tohto súboru zahŕňajú napríklad verziu aplikácie, použité prístupové oprávnenia alebo komponenty z ktorých sa aplikácia skladá

- Informácie o certifikáte – dáta získané analýzou súboru *CERT.RSA* v priečinku *META-INF*(viď 3.1), napríklad použitý algoritmus podpisovania, názov vydavateľa alebo MD5 hash celého certifikátu. Pred prístupom k súboru je *CERT.RSA* je nutné APK balíček rozbaliť

- Informácie o zdrojových súboroch[1] – informácie o zdrojoch aplikácie, ako napríklad formát alebo veľkosť obrázkových súborov, počet lokalizácií aplikácie alebo počet surových neskomprimovaných zdrojových súborov

- Súbory obsiahnuté v APK balíčku – zoznam všetkých súborov rozdelený do kategórií: obrázky(súbory z priečinku *res/drawable*), návrhy obrazoviek(súbory z priečinku *res/layout*), *classes.dex*, *resources.arsc* a ostatné. O každom súbore si uchovávame jeho relatívnu cestu v APK balíčku a SHA1 hash. Ako zdroj informácií slúži súbor *MANIFEST.MF* (viď 3.1)

Kompletný zoznam zbieraných metadát sa nachádza v prílohe TODO.

1. angl. resources

6 Štatistiky

Analýzou jednotlivých APK súborov získame detailné informácie o jednotlivých aplikáciách. Pre ucelenejší pohľad na všeobecné vlastnosti a atribúty Android aplikácií je vhodné rozšíriť analýzu jednotlivých aplikácií na skúmanie väčšej množiny APK balíčkov. Keďže databáza APK súborov použitá v tejto práci obsahuje dostatočne veľkú vzorku približne 20000 APK súborov, ktoré pochádzajú z rôznych oficiálnych aj alternatívnych zdrojov, poskytuje dobrú vzorku na určenie štatistických údajov a Android aplikáciách. Štatistické informácie prezentované v tejto kapitole sa viažu k aplikáciám dostupným v rokoch 2014–2016 a teda sú aktuálne pre spomenuté obdobie. Vyvinutá aplikácia *ApkAnalyzer* poskytuje možnosť výpočtu štatistík nad množinou APK súborov. Funkcionalita výpočtu štatistických informácií sa aktivuje pomocou prepínača –statistics pri spustení programu z príkazového riadku. Ako vstup aplikácie slúžia JSON súbory vytvorené analýzou. Výstupom je súbor vo formáte JSON obsahujúci vypočítané štatistické dáta. Pri vlastnostiach, ktorých hodnota je vyjadrená číselne sú vypočítané základné matematické štatistiky ako je aritmetický priemer, modus, medián, rozptyl, smerodajná odchýlka, minimum a maximum. Pri najnižšej a najvyššej hodnote obsahuje výstup aj názov aplikácií, ktoré tieto hodnoty dosahujú. V prípade vlastností, ktoré nadobúdajú obmedzený počet predom definovaných hodnôt je určené percentuálne zastúpenie jednotlivých hodnôt.

6.1 Získané dáta

Veľkosť APK súborov

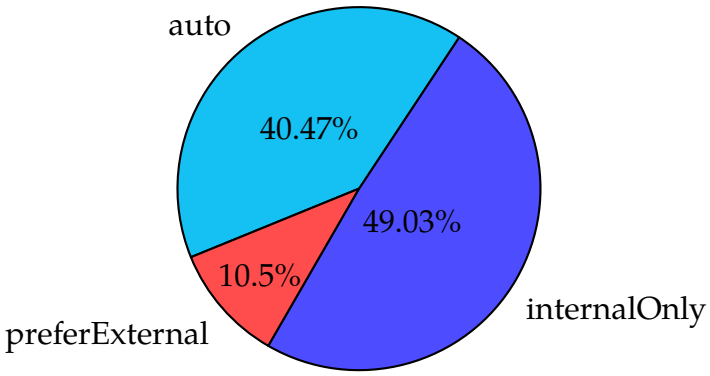
Analýzou databázy APK súborov sa zistilo, že stredná hodnota veľkosti APK súborov je 5,26 MB, priemer dosahuje hodnotu 10.19 MB.

Počet súborov v APK balíku

Strednou hodnotou celkového počtu súborov v APK balíčku je 397, aritmetický priemer má hodnotu približne 730 súborov.

Inštalačná politika

Android poskytuje možnosť špecifikovať, či aplikácia môže byť presunutá na externé dátové úložisko(viď 3.7.1). Až 49 % aplikácií neumožňuje inštaláciu alebo presun na externé pamäťové médiá. 40 % aplikácií preferuje inštaláciu na interné úložisko s možnosťou presunu do externej pamäte. 10,5 % aplikácií uprednostňuje svoju inštaláciu na externé pamäťové médium. Rozdelenie hodnôt je zobrazené v grafe 7.1.



Obr. 6.1: Hodnoty atribútu installLocation

Komponenty aplikácií

Základnou funkčnou jednotkou Android aplikácií sú aktivity (viď 3.7.7). Stredná hodnota počtu aktivít medzi analyzovanými aplikáciami je 10, priemer dosahuje hodnotu 20,23. Aplikácie obsahujú najčastejšie 2 aktivity. Priemerný počet služieb definovaných v aplikácií je 3,99, stredná hodnota je 1, no najčastejším prípadom je, že aplikácia nedefinuje žiadnu službu.

Verzie Android SDK

Najčastejšou najnižšou vyžadovanou verziou Android SDK je verzia 9 s 21,3% zastúpením. Naniższie vyžadované verzie Android SDK v

našej databáze APK súborov sú zobrazené v grafe A.4. Až 25,64 % aplikácií je primárne určených na SDK verziu 19.

Prístupové oprávnenia

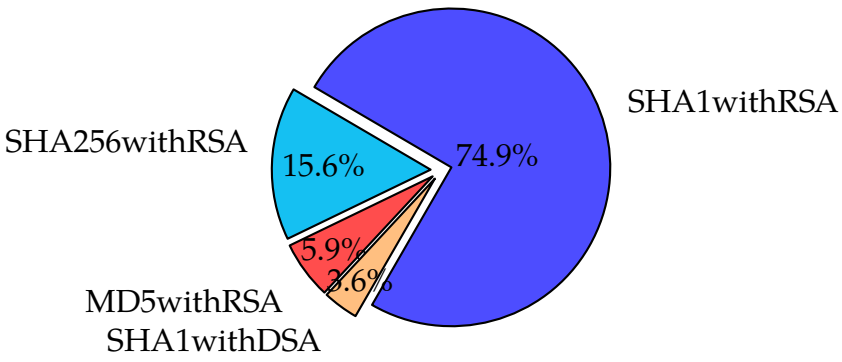
Android aplikácie najčastejšie deklarujú, že využívajú 4 prístupové oprávnenia (viď 3.7.2). Stredná hodnota počtu vyžadovaných oprávnení je 8. 10 najčastejšie využívaných oprávnení spolu s ich percentuálnym zastúpením v analyzovanej vzorke aplikácií je uvádených v tabuľke A.3.

Využité vlastnosti

Aplikácie deklarujú nízky počet využívaných vlastností (viď 3.7.2). Aritmetický priemer je 1,44, stredná hodnota a modus sú nulové. Najčastejšie deklarované je využívanie vlastností uvedených v tabľke A.2.

6.1.1 Podpis APK balíčka

Na podpisovanie APK balíčkov je v najčastejšie využitý algoritmus *SHA1withRSA*, ktorý využíva až 74,87 % aplikácií. 15,56 % aplikácií je podpísaných pomocou algoritmu *SHA256withRSA*, podpis pomocou *MD5withRSA* je využitý v 5,88 % prípadoch.



Obr. 6.2: Algoritmus podpisu APK balíčku

Lokalizácia

Aplikácie sú okrem základného jazyka lokalizované najčastejšie v 17 iných jazykoch. Aritmetický priemer počtu lokalizácií je 31,42. Najčastejšie lokalizácie aplikácií sú uvedené v tabuľke A.1. Lokalizácie sú ovplyvnené tým, že časť aplikácií bola stiahnutá z portálov určených pre strednú európu. V českom jazyku je lokalizovaných 49 % aplikácií, v slovenčine 46 %.

Obrázkové súbory

Analýza ukázala, že aplikácie využívajú množstvo obrázkových súborov. Stredná hodnota celkového počtu obrázkových súborov je 210, aritmetický priemer dosahuje hodnotu 462 a najčastejším počtom obrázkových súborov je 5 . Stredná hodnota počtu rozdielnych obrázkov (veľkosť a rozlíšenie sa neberie do úvahy) je 134. Najčastejším formátom je PNG, aplikácia obsahuje priemerne 342,8 takýchto súborov.

7 Prebalené APK súbory

http://www.csc.ncsu.edu/faculty/jiang/pubs/CODASPY12.pdf Pojem prebalený súbor označuje APK balíčky, ktoré boli modifikované no navonok sa prezentujú ako originálne neupravené aplikácie. Častým prípadom je, že takéto aplikácie pochádzajú z oficiálneho zdroja Android aplikácií – Google Play Strore, sú upravené a následne redistribuované pomocou neoficiálnych zdrojov. Takéto aplikácie si spravidla ponechávajú dizajn a funkcionalitu originálnych aplikácií, ku ktorej však môžu pridávať nové neželané funkcie alebo modifikácie. Hlavnou motiváciou pri modifikovaní aplikácií je šírenie škodlivého softvéru – malvéru. Pozmenená aplikácia môže napríklad získať prístup k citlivým informáciám uložených v Android zariadení alebo monitorovať správanie užívateľa. Modifikovaná aplikácia môže obsahovať nové reklamy. Prebalené aplikácie majú negatívny vplyv na vývojárov originálnej aplikácie. V prípade, že aplikácia obsahuje možnosť nákupov priamo z aplikácie[1], výnosy z týchto predajov môžu byť presmerované z účtov originálnych vývojárov na účet ľudí, ktorí aplikáciu modifikovali. Ovplyvnené sú aj obchody s aplikáciami na ktorých sa nachádzajú prebalené aplikácie, keďže používatelia uprednostnia kvalitnejšie zdroje. V súčasnosti žiadny z obchodov s aplikáciami vrátane oficiálneho Google Play nepoužíva efektívnu detekciu prebalených aplikácií. http://www.cs.columbia.edu/ nieh/pubs/sigmetrics2014_playdrone.pdf

7.1 Modifikácia APK súborov

Modifikácia APK súborov nie je náročná. Aplikácia môže byť jednoducho rozbalená a zdrojové súbory ako napríklad obrázky môžu byť upravené alebo nahradené inými. Štrukturovaný proces tvory APK (http://www.alittlemadness.com/2010/06/07/understanding-the-android-build-process/) balíčkov umožňuje jednoduchú dekompiláciu. V prípade modifikácie zdrojového kódu je možné použiť existujúce nástroje ako *dex2Jar* alebo *ApkTool*, ktoré sú bližšie popísané v kapitole 5.1. Modifikácia súboru *AndroidManifest.xml* je takisto možná.

1. angl. in-app purchase

Pomocou existujúcich nástrojov je ho možné previesť do čitateľného XML súboru, ktorý je možné editovať a následne previesť späť do binárneho XML formátu. Túto funkcionalitu poskytuje okrem iných utilít aj ApkTool. Android obsahuje ochranu pred narušením integrity APK balíčka, ktorá je zabezpečená pomocou súborov v priečinku META-INF v koreňovej zložke APK súboru. Informácie o týchto súboroch sa nachádzajú v kapitole 3.1. V prípade detekcii narušenia integrity, Android zakáže inštaláciu danej aplikácie. Po každej zmene súborov v APK balíčku je nutné podpísať ho. Aplikácie sú zvyčajne podpísané certifikátom ktorý je podpísaný identitou ktorú identifikuje[2]. Vďaka tomu je možné po modifikácií APK balíček podpísať a zabezpečiť tak jeho fungovanie.

7.2 Známe metódy detekcie prebalených APK súborov

Jednoduchá modifikácia inštalačných balíčkov predstavuje problém pre celkový ekosystém aplikácií pre Android. Riešenie problému pozmeňovania a redistribúcie APK súborov je v súčasnosti dôležitou témou. Bolo navrhnutých viacero spôsobov detekcie prebalených aplikácií.

7.2.1 Detekcia pomocou analýzy zdrojového kódu

Väčšina navrhnutých spôsobov detekcie modifikovaných APK balíčkov využíva metódu analyzujúcu zdrojový kód aplikácie spolu so súborom AndroidManifest.xml. Úprava zdrojového kódu je nevyhnutná v prípade editácie za účelom pridania novej neželanej funkcionality, pridania nových knižníc s reklamou alebo editácie pôvodnej reklamy použitej v aplikácii. Motiváciou pre editáciu metasúboru AndroidManifest.xml je možnosť pridávať aplikáciám prístupové povolenia (viď kapitola 3.7.2). Riešenia sú založené na použití statickej analýzy kódu, dynamickej analýzy alebo na detekcií známych vzoriek škodlivého kódu[3]. Takéto riešenia boli prezentované v prácach TODO.

2. angl. self-signed certificate
3. angl. signature based

Huang, H., Zhu, S., Liu, P., Wu, D.: A framework for evaluating mobile app repackaging detection algorithms. In: Proc. of TRUST '13. pp. 169–186 (2013) Crussell, J., Gibler, C., Chen, H.: Attack of the clones: Detecting cloned applications on android markets. In: Proc. of ESORICS'12. pp. 37–54 (2012) Crussell, J., Gibler, C., Chen, H.: Scalable semantics-based detection of similar android applications. In: Proc. of Esorics'13 (2013) Gibler, C., Stevens, R., Crussell, J., Chen, H., Zang, H., Choi, H.: Adrob: examining the landscape and impact of android application plagiarism. In: Proc. of MobiSys'13. pp. 431–444 (2013) Hanna, S., Huang, L., Wu, E., Li, S., Chen, C., Song, D.: Juxtapp: a scalable system for detecting code reuse among android applications. In: Proc. of DIMVA'12. pp. 62–81 (2013) Zhou, W., Zhou, Y., Jiang, X., Ning, P.: Detecting repackaged smartphone applications in third-party android marketplaces. In: Proc. of CODASPY'12. pp. 317–326 (2012) Potharaju, R., Newell, A., Nita-Rotaru, C., Zhang, X.: Plagiarizing smartphone applications: attack strategies and defense techniques. In: Proc. of ESSoS'12. pp. 106–120 (2012)

7.2.2 Detekcia pomocou podobnosti súborov

Repackaged APK súbory je možné úspešne detekovať prostredníctvom zhody súborov obsiahnutých v APK balíčku. Tento prístup prezentoval Yury Zhauniarovich v svojej práci *FSquaDRA: Fast Detection of Repackaged Applications* (http://www.zhauniarovich.com/files/pubs/Fsquadra_Zhauniarovi , https://github.com/zyrikby/FSquaDRA). Prístup využíva skutočnosť, že aplikácia nie je definovaná iba svojim zdrojovým kódom a funkcionalitou, ale je tvorená aj ďalšími dôležitými prvkami ako sú používateľské prostredie alebo multimediálny obsah. APK balíčky obsahujú množstvo doplnkových zdrojových súborov. Základom tohto prístupu je pozorovanie, že modifikované aplikácie zachovávajú užívateľské rozhranie ,dizajn, ikony, obrázky alebo zvuky pôvodných aplikácií. Práve tieto prvky výrazne odlišujú aplikácie, identifikujú ich pre užívateľov a majú výrazný dopad na užívateľský dojem. Preto je veľká časť súborov z neupravených APK balíčkoch obsiahnutá aj v modifikovaných balíčkoch. Originálna aplikácia Opera Mini a verzia tejto aplikácie obsahujúca malware (http://www.zdnet.com/ warning-new-android-malware-tricks-users-with-real-opera-mini-7000001586/), sa zhodujú v 230 z 234 súborov nachádzajúcich sa v príslušných APK balíčkoch.

Riešenie prezentované v práci *FSquaDRA* porovnáva všetky súbory medzi dvoma APK balíčkami. Porovnávanie jednotlivých súborov na binárnej úrovni by bolo výpočtovo náročné. Preto sa na porovnanie využívajú SHA1 hashe súborov, ktoré sa nachádzajú v súbore *MANIFEST.MF* (viď 3.1). Podobnosť aplikácií je určená na základe Jaccard indexu. V práci sa rozlišujú dva typy podobných APK súborov. Aplikácie sú považované za plagiátorsky prebalené aplikácie, keď obsahujú mnoho rovnakých súborov, ale sú podpísané rôznymi certifikátmi. V prípade veľkej zhody súborov a identických certifikátov, sú aplikácie považované za rôzne verzie jednej aplikácie a nie sú označené ako nebezpečné. Tento spôsob porovnávania neumožňuje určiť, ktorá z aplikácií je originálna, a ktorá je pozmenená. Umožňuje však rýchlu a efektívnu detekciu modifikovaných APK súborov.

7.3 Navrhnutá metóda detekcie prebalených APK súborov

Spôsob detekcie pozmenených APK súborov prezentovaný v rámci tejto práce vychádza zo základných metód prezentovaných v článku *FSquaDRA: Fast Detection of Repackaged Applications*. Prístup je založení na podobnosti súborov. Celková podobnosť aplikácií je určená podľa počtu zhodných súborov prítomných v oboch APK balíčkoch. Účelom našej implementácie nie je simulovať detekciu prebalených inštalačných súborov pomocou metódy *FSquaDRA*. Cieľom je implementovať program, ktorý na detekciu modifikovaných APK balíčkov používa podobnosť obsahu balíčkov, ktorú kombinuje s metadátami a informáciami o daných APK súborov. Metadáta sú využívané na zefektívnenie výpočtu, ktoré je dosiahnuté neporovnávaním súborov medzi dvojicami zjavne odlišných aplikácií. Informácie získané porovnávaním a analýzou dvoch podobných aplikácií sú užívateľovi prezentované ako výstup porovnania. V prípade podobnosti aplikácií je výstupom porovnania zoznam odlišností, a typ podobnosti dvoch aplikácií. Typ podobnosti je určený na základe zhody certifikátov a zhody verzií aplikácií.

7.3.1 Implementácia

Funkcionalita porovnávania a detekcie modifikovaných APK balíčkov je implementovaná v programe *ApkAnalyzer*. *ApkAnalyzer* je aplikácia napísaná v jazyku Java, ktorá neobsahuje grafické užívateľské rozhranie. Používateľ môže aplikáciu spúšťať a zadávať mu parametre pomocou príkazového riadku. Funkcionalita porovnávania APK súborov sa spúšťa pomocou parametra *–compare*. Vstup pre porovnávanie APK súborov nie sú samotné APK balíčky, ale JSON súbory obsahujúce informácie o aplikáciách, ktoré sú vytvorené aplikáciou ApkAnalyzer počas analýzy APK balíčkov (viď kapitola 5). Analýza a následné porovnanie môžu byť spustené sériovo. Samotné porovnávanie prebieha párovo, každá aplikácia je porovnávaná so všetkými ostatnými. Proces porovnávania je paralelizovaný a každé dostupné procesorové jadro porovnáva inú dvojicu aplikácií. Porovnávanie a vyhodnocovanie podobnosti je rozdelené do viacerých etáp. Najskôr sa porovnávajú základné informácie o APK súboroch a príslušných aplikáciách. Toto porovnanie využíva základné metadata o aplikáciách a zahŕňa veľkosť APK súboru, počet komponent z ktorých sa aplikácia skladá (aktivity, služby, poskytovatelia obsahu, prijímače), počet rôznych obrázkových súborov a počet súborov definujúcich vzhľad obrazoviek(ang. layout). Všetky tieto hodnoty sú číselné. Je nutné aby implementácia ich porovnávania bola funkčná pre malé aj veľké hodnoty. Taktiež je nutné zabezpečiť komutatívnosť, ktorá zaručí, že nezáleží na poradí porovnávania aplikácií a teda func(A,B) = func(B,A). Získané hodnoty sú porovnávane s minimálnymi hodnotami potrebnými na to, aby boli aplikácie považované za podobné. Tieto hodnoty je možné meniť editáciou súboru similarity.properties v koreňovej zložke projektu *ApkAnalyzer*. V prípade detekcie základnej podobnosti sa porovnajú všetky súbory v APK balíčkoch. Podobne ako v aplikácií vyvinutej v rámci projektu *FSquaDRA*, porovnávajú sa SHA1 hashe súborov uložené v *MANIFEST.MF*. Separátne sú porovnávané súbory *classes.dex*, *resources.arsc*. Ostatné súbory sú porovnávané v rámci kategórií : súbory v priečinku drawable, súbory v priečinku layout, ostatné súbory, všetky súbory. Zhoda súborov medzi dvomi APK balíčkami je určená pomocou Jaccard index(https://en.wikipedia.org/wiki/Jaccard_index). Nech A sú súbory v danej kategórií v jednom APK balíčku, a B sú súbory v danej kategórií v druhom porovnávanom APK balíčku. Jaccard

Index(A,B) = A prienik B / A zjednotenie B. Aplikácie sú považované za podobné v prípade, že Jaccard index pre každú z kategórií prekračuje minimálnu hodnotu definovanú v súbore *similarity.properties*. Okrem súborov sa porovnajú aj všetky hodnoty získané analýzou APK súboru, no určenie podobnosti v tejto fáze prebieha len na základe rovnakých súborov.

Typ podobnosti
Zhoda certifikátov a zhoda verzií aplikácií je vypočítaná za účelom určenia typu podobnosti daných aplikácií. Pre zhodu verzií a certifikátov rozlišuje tri hodnoty – rovnaké, rozdielne alebo neurčené. Hodnota neurčené je použitá v prípade, že sa dáta nepodarilo získať. Zhoda certifikátov sa určuje na základe MD5 hashu certifikátu. Tento údaj je v kontexte detekcie modifikovaných APK súborov veľmi dôležitý. V prípade zhody certifikátov je zaručené, že APK súbory pochádzajú od rovnakého vydavateľa. Pokiaľ sú certifikáty rozdielne, pôvodca súborov je z najväčšou pravdepodobnosťou rozdielny. Zhoda verzií aplikácií je využitá na detekciu rovnakých aplikácií v rozdielnych verziách. Kombinácia týchto hodnôt určuje 9 kategórií podobnosti APK súborov. Každá z týchto kategórií napovedá a vzájomnom vzťahu danej dvojice Android aplikácií. Najväčšia pravdepodobnosť, že aplikácia je prebalená je v prípade rovnakých verzií a zároveň rozdielnych certifikátov.

Výstup porovnania
V prípade, že porovnávaná dvojica APK súborov je vyhodnotená ako podobná, *ApkAnalyzer* vytvorí výstupný súbor vo formáte JSON obsahujúci rozdiely medzi danými aplikáciami. Tento súbor obsahuje rozdiely určené na základe metadát a porovnania aplikácií. Slúži ako jednoduchá obdoba linuxového príkazu diff implementovaná nad APK súbormi. Obsahuje informácie o modifikovaných parametroch a komponentoch aplikácií a taktiež zoznam upravených, nových alebo odstránených súborov. Príklad takéhoto výstupného súboru sa nachádza v prílohe TODO.

8 These are

8.1 the available

8.1.1 sectioning commands.

Paragraphs and

subparagraphs are available as well. Inside the text, you can also use unnumbered lists,

- such as
- this one
  - and they can be nested as well.
  - » You can even turn the bullets into something fancier,
  - § if you so desire.

Numbered lists are

1. very
   (a) similar

and so are description lists:

Description list A list of terms with a description of each term

The spacing of these lists is geared towards paragraphs of text. For lists of words and phrases, the paralist package offers commands

- that
  - are
    - ∗ better
      - · suited

1. to
   (a) this
      i. kind of
         A. content.

9 Floats and references

The logo of the Masaryk University is shown in Figure 10.1 and Figure 10.2 at pages 47 and 48. The weather forecast is shown in Table 10.1 at page 48. The following chapter is Chapter 11 and starts at page 49. Items 3, 3b, and 3(c)iv are starred in the following list:

1. some text
2. some other text
3. ⋆
   (a) some text
   (b) ⋆
   (c) some other text
      i. some text
      ii. some other text
      iii. yet another piece of text
      iv. ⋆
   (d) yet another piece of text
4. yet another piece of text

If your reference points to a place that has not yet been typeset, the `\ref` command will expand to **??** during the first run of `pdflatex thesis.tex` and a second run is going to be needed for the references to resolve. With online services – such as Overleaf – this is performed automatically.

Obr. 9.1: The logo of the Masaryk University at 40 mm

Obr. 9.2: The logo of the Masaryk University at $\frac{2}{3}$ and $\frac{1}{3}$ of text width

Day	Min Temp	Max Temp	Summary
Monday	13°C	21°C	A clear day with low wind and no adverse current advisories.
Tuesday	11°C	17°C	A trough of low pressure will come from the northwest.
Wednesday	10°C	21°C	Rain will spread to all parts during the morning.

Tabuľka 9.1: A weather forecast

10 Mathematical equations

TEX comes pre-packed with the ability to typeset inline equations, such as $e^{ix} = \cos x + i \sin x$, and display equations, such as

$$\mathbf{A}^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{\det(\mathbf{A})} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}.$$

LATEX defines the automatically numbered `equation` environment:

$$\gamma Px = PAx = PAP^{-1}Px. \tag{10.1}$$

The package amsmath provides several additional environments that can be used to typeset complex equations:

1. An equation can be spread over multiple lines using the `multline` environment:

$$\begin{aligned} a + b + c + d + e + f + b + c + d + e + f + b + c + d + e + f \\ + f + g + h + i + j + k + l + m + n + o + p + q \end{aligned} \tag{10.2}$$

2. Several aligned equations can be typeset using the `align` environment:

$$a + b = c + d \tag{10.3}$$
$$u = v + w + x \tag{10.4}$$
$$i + j + k + l = m \tag{10.5}$$

3. The `alignat` environment is similar to `align`, but it doesn't insert horizontal spaces between the individual columns:

$$a + b + c + d = 0 \tag{10.6}$$
$$e + f + g = 5 \tag{10.7}$$

4. Much like chapter, sections, tables, figures, or list items, equations – such as (11.8) and (My equation) – can also be labeled and referenced:

$$b_{11}x_1 + b_{12}x_2 + b_{13}x_3 = y_1, \tag{10.8}$$
$$b_{21}x_1 + b_{22}x_2 + b_{24}x_4 = y_2. \tag{My equation}$$

5. The `gather` environment makes it possible to typeset several equations without any alignment:

$$\psi = \psi\psi, \tag{10.9}$$
$$\eta = \eta\eta\eta\eta\eta, \tag{10.10}$$
$$\theta = \theta. \tag{10.11}$$

6. Several cases can be typeset using the `cases` environment:

$$|y| = \begin{cases} y & \text{if } z \geq 0, \\ -y & \text{otherwise.} \end{cases} \tag{10.12}$$

For the complete list of environments and commands, consult the amsmath package manual[1].

1. See `http://mirrors.ctan.org/macros/latex/required/amslatex/math/amsldoc.pdf`. The `\url` command is provided by the package url.

11 We have several FONTS *at* **disposal**

The serified roman font is used for the main body of the text. *Italics are typically used to denote emphasis or quotations.* `The teletype font is typically used for source code listings.` The **bold**, SMALL-CAPS and sans-serif variants of the base roman font can be used to denote specific types of information.

We can also change the font size, although it is usually not necessary.

A wide variety of mathematical fonts is also available, such as:

$$\mathrm{ABC}, \mathcal{ABC}, \mathbf{ABC}, \mathsf{ABC}, \mathit{ABC}, \mathtt{ABC}$$

By loading the amsfonts packages, several additional fonts will become available:

$$\mathfrak{ABC}, \mathbb{ABC}$$

Many other mathematical fonts are available[1].

1. See `http://tex.stackexchange.com/a/58124/70941`.

12 Inserting the bibliography

After loading the `biblatex` package and linking a bibliography database file to the document using the `\addbibresource` command, you can start citing the entries. This is just dummy text [**inbook-full**] lightly sprinkled with citations [**incollection-full**]. Several sources can be cited at once [**whole-collection**, **manual-minimal**, **manual-full**]. **inbook-full** was written by **inbook-full** in **inbook-full** We can also produce **inbook-full** or (**inbook-full**, **inbook-full**). The full bibliographic citation is: **inbook-full**. We can easily insert a bibliographic citation into the footnote[1].

The `\nocite` command will not generate any output, but it will insert its argument into the bibliography. The `\nocite{*}` command will insert all the records in the bibliography database file into the bibliography. Try uncommenting the command and watch the bibliography section come apart at the seams.

When typesetting the document for the first time, citing a `work` will expand to [**work**] and the `\printbibliography` command will produce no output. It is now necessary to generate the bibliography by running `biber thesis.bcf` from the command line and then by typesetting the document again twice. During the first run, the bibliography section and the citations will be typeset, and in the second run, the bibliography section will appear in the table of contents.

The `biber` command needs to be executed from within the directory, where the LaTeX source file is located. In Windows, the command line can be opened in a directory by holding down the Shift key and by clicking the right mouse button while hovering the cursor over a directory. Select the Open Command Window Here option in the context menu that opens shortly afterwards.

With online services – such as Overleaf – all commands are executed automatically.

1. **inbook-full**.

Literatúra

[1] Jimmy Westenberg. *Gartner: Android and iOS dominate smartphone market with 98 percent marketshare*. 2015. URL: http://www.androidauthority.com/android-ios-hold-98-percent-marketshare-656624/.

13 Inserting the index

After using the `\makeindex` macro and loading the `makeidx` package that provides additional indexing commands, index entries can be created by issuing the `\index` command. It is possible to create ranged index entries, which will encompass a span of text. To insert complex typographic material – such as α or TEX – into the index, you need to specify a text string, which will determine how the entry will be sorted. It is also possible to create hierarchal entries.

After typesetting the document, it is necessary to generate the index by running

`texindy -I latex -C utf8 -L` ⟨*locale*⟩ `thesis.idx`

from the command line, where ⟨*locale*⟩ corresponds to the main locale of your thesis – such as `english`, and then typesetting the document again.

The `texindy` command needs to be executed from within the directory, where the LATEX source file is located. In Windows, the command line can be opened in a directory by holding down the Shift key and by clicking the right mouse button while hovering the cursor over a directory. Select the Open Command Window Here option in the context menu that opens shortly afterwards.

With online services – such as Overleaf – the commands are executed automatically, although the locale may be erroneously detected, or the `makeindex` tool (which is only able to sort entries that contain digits and letters of the English alphabet) may be used instead of `texindy`. In either case, the index will be ill-sorted.

A An appendix

Kód	Jazyk	%
es	španielsky	61,7
de	nemecký	59,6
fr	francúzsky	59,4
ru	ruský	58,1
ja	japonský	57,6
it	taliansky	57,4
ko	korejský	56,9
zh-rcn	čínsky (zjednodušený)	55,6
zh-rtw	čínsky (tradičný)	54,0
pt	portugalský	52,6

Tabuľka A.1: Lokalizácia aplikácií

Názov	%
android.hardware.camera	18,1
android.hardware.touchscreen	16,1
android.hardware.telephony	14,8
android.hardware.camera.autofocus	10,6
android.hardware.location.gps	10,2
android.hardware.location	8,8
android.hardware.wifi	8,4
android.hardware.location.network	7,0
android.hardware.bluetooth	6,6
android.hardware.touchscreen.multitouch	6,0

Tabuľka A.2: Najpoužívanejšie vlastnosti

Názov	%
android.permission.internet	92,9
android.permission.access_network_state	87,9
android.permission.write_external_storage	75,2
android.permission.wake_lock	49,5
android.permission.read_phone_state	49,4
android.permission.access_wifi_state	44,7
android.permission.vibrate	43,6
android.permission.get_accounts	31,3
android.permission.receive_boot_completed	30,5
android.permission.vending.billing	27,1

Tabuľka A.3: Najpoužívanejšie prístupové oprávnenia

Verzia Android SDK	%
9	21,3
8	18,4
7	14,2
14	10,5
10	8,1
4	7,0
3	5,6
15	3,7
5	3,7
11	2,1

Tabuľka A.4: Hodnoty najnižsej vyžadovanej verzie Android SDK

Verzia Android SDK	%
19	25,6
17	11,8
21	11,7
15	6,8
14	6,3
22	6,0
16	5,7
18	5,6
20	3,8
8	2,9

Tabuľka A.5: Hodnoty cieľovej verzie Android SDK