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 obsiahnutých.

# Kľúčové slová

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

# Obsah


1 **Úvod** . . . . . 1
2 **Operačný systém Android** . . . . . 3
2.1 *História* . . . . . 3
2.2 *Architektúra systému* . . . . . 4
2.2.1 Linuxové jadro . . . . . 4
2.2.2 Android Runtime a Dalvik Virtual Machine . . . 4
2.2.3 Knižnice . . . . . 5
2.2.4 Aplikačný rámec . . . . . 5
2.2.5 Vrstva aplikácií . . . . . 6
2.3 *Aplikácie* . . . . . 6
2.3.1 Distribúcia aplikácií . . . . . 6
2.3.2 Inštalácia aplikácií . . . . . 6
3 **APK súbory** . . . . . 9
3.1 *Priečinok META-INF* . . . . . 9
3.2 *Priečinok res* . . . . . 11
3.3 *Priečinok lib* . . . . . 13
3.4 *Priečinok assets* . . . . . 13
3.5 *resources.arsc* . . . . . 14
3.6 *classes.dex* . . . . . 14
3.7 *AndroidManifest.xml* . . . . . 14
3.7.1 Element manifest . . . . . 15
3.7.2 Element uses-permission . . . . . 16
3.7.3 Element permission . . . . . 17
3.7.4 Element uses-sdk . . . . . 18
3.7.5 Element uses-feature . . . . . 19
3.7.6 Element supports-screens . . . . . 20
3.7.7 Element activity . . . . . 20
3.7.8 Element service . . . . . 21
3.7.9 Element provider . . . . . 21
3.7.10 Element receiver . . . . . 21
3.7.11 Element uses-library . . . . . 22
4 **Databáza inštalačných APK súborov** . . . . . 23
4.1 *Implementácia* . . . . . 24
5 **Analýza APK súborov** . . . . . 25
5.1 *Nástroje reverzného inžinierstva* . . . . . 25

5.1.1 ApkTool . . . . . 25
5.1.2 Dex2Jar . . . . . 25
5.1.3 AAPT . . . . . 26
5.1.4 AXML . . . . . 26
5.2 *Implementácia analýzy* . . . . . 26
6 **Štatistiky** . . . . . 29
6.1 *Získané dáta* . . . . . 29
7 **Prebalené APK súbory** . . . . . 37
7.1 *Modifikácia APK súborov* . . . . . 37
7.2 *Známe metódy detekcie prebalených APK súborov* . . . . . 38
7.2.1 Detekcia pomocou analýzy zdrojového kódu . . 38
7.2.2 Detekcia pomocou podobnosti súborov . . . . . 39
7.3 *Navrhnutá metóda detekcie prebalených APK súborov* . . . . 39
7.3.1 Implementácia . . . . . 41
7.3.2 Výsledky . . . . . 43
8 **Záver** . . . . . 45
A **Prílohy** . . . . . 47

# 1 Úvod

Chytré mobilné zariadenia sú v súčasnosti veľmi obľúbené a rozšírené. Na trhu s mobilnými telefónmi dominujú smartfóny, ktoré neplnia len funkciu telefónu, ale je v nich integrovaná aj funkcionalita fotoaparátu, multimediálneho prehrávača alebo GPS navigácie. Chytré zariadenia dnes obsahujú mnoho funkcií, na ktoré bol kedysi potrebný špecializovaný hardvér. Smartfóny a tablety obsahujú plnohodnotný operačný systém a poskytujú užívateľom možnosť výberu z množstva aplikácií, ktoré môžu využívanie chytrého zariadenia zefektívniť alebo spríjemniť. Na trhu s mobilnými operačnými systémami je dominantný systém Android. Užívatelia preferujú Android kvôli prívetivému užívateľskému rozhraniu a dobrej dostupnosti aplikácií. Medzi výrobcami je tento systém populárny vďaka voľne dostupnému zdrojovému kódu a môžnosti modifikácie a vyladenia pre potreby konkrétneho zariadenia.

V tejto práci sa zaoberám získavaním a analýzou metainformácií o inštalačných súboroch a aplikáciách pre mobilný operačný systém Android. Aplikácie pre Android sú distribuované formou inštalačných APK balíčkov. V rámci práce je vytvorená databáza takýchto inštalačných súborov.

Cieľom práce je analýza APK súborov za účelom získania metadát. Zámerom je získať metainformácie o veľkom počte vlastností z rôznych častí APK súboru. Metainformácie obsahujú údaje o veľkosti aplikácií, počte a formáte multimediálnych súborov, certifikáte, komponentách a preferenciách aplikácie, ale aj o súboroch obsiahnutých v APK balíčku. Na základe týchto metadát sú na vzorke aplikácií z vytvorenej databázy určené štatistické vlastnosti APK súborov.

Hlavným dôvodom analýzy APK súborov je však bezpečnosť samotných aplikácií a s tým súvisiaca bezpečnosť celého systému Android. Metadáta získané analýzou sú použité pri detekcii potenciálne škodlivých APK balíčkov. V súčasnosti existuje viacero metód detekcie škodlivých modifikovaných APK súborov. V tejto práci by som rád nadviazal na metódu založenú na podobnosti súborov obsiahnutých v APK balíčkoch. Primárnym cieľom pri detekcii modifikovaných APK súborov je využitie metadát na efektívne odhalenie aplikácií, ktoré boli pozmenené. Narozdiel od ostatných metód detekcie modifikova-

ných APK súborov je mojim cieľom poskytnúť užívateľovi detailný výstup obsahujúci rozdiely medzi dvojicou podobných APK balíčkov.

Výsledok tejto práce je databáza APK súborov obsahujúca aplikácie pochádzajúce primárne z alternatívnych zdrojov. V rámci práci je implementovaný nástroj na automatizované sťahovanie APK súborov. Ďalším výstupom je aplikácia *ApkAnalyzer*, ktorá slúži na hromadnú analýzu APK súborov. Práca obsahuje štatistické dáta získané analýzou inštalačných súborov. V práci je popísaný navrhnutý spôsob detekcie potenciálne škodlivých prebalených APK súborov. Navrhnutá metóda je prakticky implementovaná v aplikácii *ApkAnalyzer*.

# 2 Operačný systém Android

Android je mobilný operačný systém navrhnutý primárne pre zariadenia s dotykovou obrazovkou. Android je dominantným operačným systémom na trhu s mobilnými zariadeniami ako sú chytré telefóny a tablety. V treťom kvartáli roku 2015 dosahoval až 84,7% podiel na trhu operačných systémov pre mobilné zariadenia [**Westenberg2015**]. Systém je založený na linuxovom jadre.

Android je aktuálne vyvíjaný spoločnosťou Google ako open source projekt. Existuje aktívna komunita vývojárov podieľajúca sa na vývoji 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, tzv. Google services. Android nemá žiadny centralizovaný systém aktualizácií. To má za následok, že veľká časť zariadení nedostáva aktualizácie dostatočne často. Až 87,7 % zariadení obsahuje kritické bezpečnostné zraniteľnosti, ktoré sú známe, ale nie sú opravené kvôli slabej podpore [**Thomas2015**].

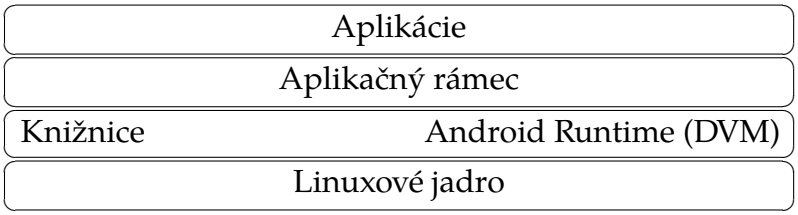
## 2.1 História

Začiatok vývoja operačného systému, ktorý je dnes známy ako Android siaha do roku 2003, kedy vznikla spoločnosť Android, Inc. Prvotným zámerom spoločnosti bola tvorba systému pre digitálne fotoaparáty, avšak kvôli malému trhu bol vývoj preorientovaný na operačný systém pre mobilné zariadenia. Zakladatelia spoločnosti Andy Rubin, Rich Miner, Nick Sears a Chris White plánovali vývoj operačného systému pre chytré mobilné zariadenia, ktoré dokážu efektívne využívať prostredie a preferencie užívateľov [**Beavis2008**]. Spoločnosť Android bola v roku 2005 kúpená spoločnosťou Google za približne 50 miliónov dolárov [**Rosoff2011**]. 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 [**OHA**]. Prvým predstaveným produktom tejto skupiny bol operačný systém založený na linuxovom jadre – Android. Prvým komerčne dostupným chytrým telefónom s operačným systé-

mom Android sa 22. novembra 2008 stal *HTC Dream*. Od roku 2008 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.

## 2.2 Architektúra systému

Operačný systém Android je možné dekomponovať do piatich sekcií a štyroch základných architektonických vrstiev organizovaných v zásobníkovej štruktúre zobrazenej na obrázku 2.1.




Obr. 2.1: Vrstevnatá architektúra systému Android

### 2.2.1 Linuxové jadro

Najnižšiu vrstvu predstavuje Linux 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. Na tejto vrstve sa nachádzajú ovládače hardvérových komponent ako fotoaparát, dotyková obrazovka alebo sieťové rozhranie [**architecture**].

### 2.2.2 Android Runtime a Dalvik Virtual Machine

*Dalvik Virtual Machine* je virtuálny stroj slúžiaci na exekúciu Android aplikácií [**dalvik**]. Je obdobou virtuálneho stroja *JVM*[1] používaného pri jazyku Java. Virtuálny stroj *Dalvik* využíva nízkoúrovňovú funkcionalitu linuxového jadra. Každá aplikácia je spustená vo vlastnom procese a na vlastnej inštancii virtuálneho stroja. Tento prístup zaručuje,

1. Java Virtual Machine

že aplikácie sa navzájom neúmyselne neovplyvňujú, nepristupujú priamo k hardvéru zariadenia a využívajú abstrakciu, ktorá zabezpečuje ich platformovú nezávislosť [**architecture**]. Od verzie Android 5.0 je virtuálny stroj *Dalvik* plne nahradený novým behovým 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++. Takéto knižnice slúžia ako obal okolo C/C++ kódu, ku ktorému sprostredkúvajú prístup pomocou jazyka Java. Táto vrstva obsahuje niektoré štandardné knižnice známe z jazyka Java, ktoré sú upravené pre využitie na operačnom systéme Android, ale aj knižnice špecifické pre platformu Android, tzv. *Android knižnice* [**architecture**].

### 2.2.4 Aplikačný rámec

Vrstva aplikačného rámca poskytuje vysokoúrovňové služby používané na manažment aplikácie. Využíva koncept Android aplikácií, ktoré sa skladajú z viacerých komponent. Kľúčové služby poskytované aplikačným rámcom sú [**architecture**]:

- *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 aplikácii zobrazovať upozornenia
- *View system* – poskytuje prvky grafického používateľského rozhrania aplikácie

- *Package Manager* – umožňuje aplikáciám získať informácie o ostatných aplikáciách nainštalovaných na zariadení
- *Telephony Manager* – umožňuje aplikáciám získať informácie o stave telefónnych služieb
- *Location Manager* – poskytuje aplikáciám informácie o polohe zariadenia

### 2.2.5 Vrstva aplikácií

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*[2]. Aplikácie môžu byť distribuované pomocou alternatívnych obchodov ako napríklad *Amazon Appstore*[3] alebo *SlideMe*[4]. Operačný systém Android v základnom 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 aplikácie. Viacej informácií o prebalených aplikáciách sa nachádza v kapitole 7.

### 2.3.2 Inštalácia aplikácií

Aplikácie môžeme rozdeliť na dve skupiny:

2. https://play.google.com/store
3. http://www.amazon.com/mobile-apps
4. http://slideme.org/

- 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 **[Elenkov2015]**:

  - 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 inštalácii 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 (systémové) 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 [**Parmar2013**].

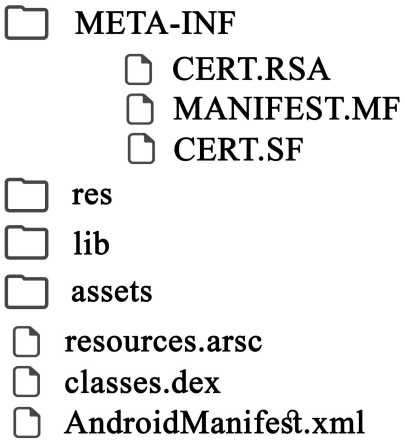
# 3 APK súbory

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 balíčkoch používaných v systéme Microsoft Windows, alebo DEB balíčkoch používaných v niektorých linuxových distribúciách. APK súbory sú asociované s príponou *.apk* a s príslušným MIME typom *application/vnd.android.package-archive* [**Freed2016**].

Štruktúra APK balíčkov vychádza z JAR (Java archive) 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 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 súbor *classes.dex* obsahujúci skompilovaný zdrojový kód, súbor *resources.arsc* ktorý obsahuje skompilované zdroje aplikácie, súbor *AndroidManifest.xml* a neskompilované súbory ako sú napríklad obrázky [**buildingAndRunning**]. Typická štruktúra APK balíčku je zobrazená na obrázku 3.1.

## 3.1 Priečinok META-INF

Priečinok *META-INF* obsahuje súbory, ktorých úlohou je zaručiť integritu ostatných súborov v APK balíčku a s ňou spojenú bezpečnosť celého systému. V prípade detekcie pozmenených súborov a narušenia integrity operačný systém Android nedovolí inštaláciu APK balíčku. Po každej zmene je nutné balíček digitálne podpísať.




Obr. 3.1: Typická štruktúra APK súboru

**CERT.RSA**

Súbor obsahujúci verejný kľúč vo forme *X509* certifikátu, ktorý slúži na overenie digitálneho podpisu balíčka.

**MANIFEST.MF**

Súbor obsahujúci relatívne cesty a SHA-1 hashe [1] všetkých súborov v APK balíčku. Tento súbor nie je špecifický len pre APK súbory, ale obsahuje ho každý JAR archív.

Typický obsah súboru *MANIFEST.MF* vyzerá nasledovne [**Yang2015**]:

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

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

Name: res/layout/activity_settings.xml
```

1. reťazce v Base64 kódovaní

```
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 textové reťazce 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* [**accessingRes**]. Všetky zdroje aplikácie by mali byť externalizované a uložené v špecifickom podpriečinku v tomto adresári.

**Podporované podpriečinky [providingRes]**

- animator - obsahuje XML súbory definujúce property animácie[2]
- anim - obsahuje XML súbory definujúce tween animácie[3], 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 - obsahuje ikony aplikácie s rôznou hustotou pixelov
- layout - obsahuje súbory vo formáte XML definujúce vzhľad a rozmiestnenie prvkov na obrazovke
- menu - obsahuje XML súbory 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 alebo 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. 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

2. Animácie definované pomocou zmeny atribútov vykreslovaných objektov
3. Animácie definované pomocou štartového bodu, koncového bodu, rotáciou a inými transformáciami vykreslovaného objektu

atribútu pre ktorý je tento priečinok určený. Je možné kombinovať viacero konfiguračných atribútov [**providingAltRes**].

**Najčastejšie používané atribúty**

- Jazyk a región – jazyk je definovaný podľa kódovania *ISO 639-1* 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 – umožňuje definovať rôzne zdrojové súbory v závislosti na orientácii zariadenia. Hodnota *port* špecifikuje súbory pre zariadenie orientované vertikálne, hodnota *land* je určená pre horizontálnu orientáciu zariadenia

- 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 a architektúru 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 umiestnenýmv 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 [**AssetManager**].

## 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 SDK* [**Reddy2014**]. 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* [**dalvik**].

## 3.7 AndroidManifest.xml

Súbor ktorý musí obsahovať každá 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 aplikáciou na prístup k chráneným častiam Android API a taktiež oprávnenia, ktoré sú vyžadované od iných komponent pri pokuse komunikovať s danou aplikáciou [**appManifest**]. 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 kapitole 6. Preto sa detailnejšie pozrieme na jeho štruktúru a niektoré dôležité elementy, ktoré obsahuje.

```
    <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 súbor *AndroidManifest.xml* ho obsahuje práve jeden.

Element *manifest* definuje atribúty **[elManifest]**:

- *xmlns:android* – povinný atribút definujúci menný priestor
- *package* – nemenný názov balíku aplikácie, povinný atribút definujúci identitu aplikácie
- *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 verzii aplikácie prezentovaná užívateľovo. Pre systém Android neposkytuje informáciu o verzii, tú obsahuje atribút *android:versionCode*.
- *android:installLocation* – určuje predvolené umiestnenie 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 sa použije v prípade, že tento atribút nie je definovaný. V prípade hodnoty *auto* sú aplikácie inštalované do 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 funkciám Andorid API je z dôvodu ochrany limitovaný. Android využíva princíp povolení[4]. 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 elementu *uses-permissions* deklarovať vyžadovaný prístup k chráneným častiam [**appManifest**]. 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 [**elUsesPerm**]:

- *android:name* – 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 poskytovanej danou aplikáciou.

4. angl. permissions

Najdôležitejšie atribúty definované v rámci elementu *permission* [**elPerm**]:

- *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. 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*, ktorý 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 aplikácii prístup k citlivým dátam alebo k prvkom 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 typu *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* [**elUsesSdk**]:

- *android:minSdkVersion* – vyjadruje najnižší level API vyžadovaný aplikáciou. Pokiaľ je táto hodnota vyššia ako API úroveň zariadenia, systém zabráni inštalácii. 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 [5] 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 [**elUsesFeature**]:

- *android:name* – názov vyžadovanej vlastnosti

5. angl. feature

- *android:required* – určuje, či aplikácia vyžaduje danú vlastnosť pre korektné fungovanie. Pokiaľ obsahuje hodnotu *true*, aplikácia nie je schopná korektne fungovať na zariadení nepodporujúcom danú vlastnosť

### 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. Medzi ostatné atribúty definované elementom *supports-screens* patria [**elScreen**]:

- *android-resizeable* – indikuje schopnosť aplikácie korektne sa prispôsobiť rôznym veľkostiam obrazoviek
- *android:anyDensity* – indikuje, že 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 grafického užívateľského rozhrania a zabezpečujú komunikáciu s užívateľom. Tieto triedy sú častou zdrojového kódu a v skompilovanej forme sa nachádzajú v súbore *classes.dex* (viď 3.6). Pri viacerých spustených aktivitách

zohráva dôležitú úlohu životný cyklus aktivity [**elActivity**]. Všetky aktivity musia byť deklarované pomocou elementu *activity*. V prípade, že deklarované nie sú, systém ich bude ignorovať a nebudú spustené.

### 3.7.8 Element service

Deklaruje komponenty aplikácie typu služba[6]. 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 [**elService**]. 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 [**boundService**]. Každá služba rozširuje základnú triedu *android.app.Service* a musí byť deklarovaná pomocou elementu *service*, inak bude ignorovaná.

### 3.7.9 Element provider

Deklaruje komponenty aplikácie, ktorých úlohou je poskytovanie štrukturovaného prístupu k dátam spravovaným aplikáciou. Poskytovatelia obsahu [7] sú implementovaní 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* [**elContentProvider**]. Systém Android 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 aplikácii prijímať a reagovať na informácie o zámere spustenia aktivity[8] aj

6. angl. service
7. angl. content providers
8. Intent

v čase, keď ostatné komponenty aplikácie nie sú spustené. Systém Android môžeme oboznámiť s existenciou komponent tohto typu pomocou elementu *receiver* v súbore *AndroidManifest.xml* alebo aj dynamicky pomocou volania *Context.registerReceiver()* v zdrojovom kóde aplikácie **[elReceiver]**.

### 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*. Balíky, ktoré sú v Android aplikáciách najpoužívanejšie, napríklad *android.app*, *android.content* alebo *android.view* sú obsiahnuté v základnej knižnici, ktorá je automaticky pripojená ku každej aplikácií a nemusí byť deklarovaná týmto elementom. Balíčky, ktoré nie sú medzi základnými knižnicami, musia byť deklarované. Tento element ovplyvňuje inštaláciu aplikácie na konkrétnom zariadení a taktiež dostupnosť aplikácie v obchode *Google Play*.

Definuje atribúty **[elUsesLib]**:

- *android:name* – názov knižnice, ktorý sa nachádza v dokumentácii príslušného použitého balíčku
- *android:required* – indikuje či aplikácia nevyhnutne vyžaduje 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 je 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 s využitím projektu *Playdrone*. V rámci tohto projektu bolo v novembri 2014 z *Google Play* stiahnutých viac ako milión aplikácií dostupných pre zariadenie *Galaxy Nexus* s operátorom *T-Mobile* [**Viennot2014**]. 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 aplikácií** | **%** |
|---|---:|---:|
| Playdrone | 8200 | 40,9 |
| www.appsapk.com | 6470 | 32,3 |
| www.apkmaniafull.com | 2870 | 14,3 |
| www.androidapksfree.com | 1030 | 5,1 |
| www.zippyshare.com | 750 | 3,7 |
| torrenty | 550 | 2,7 |
| www.uloz.to | 190 | 0,9 |
| 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 neposkytuje grafické užívateľské rozhranie, ale užívateľ môže zadávať parametre prostredníctvom príkazového riadku. Podporuje sťahovanie aplikácií získaných pomocou projektu *Playdrone* alebo z nasledujúcich neoficiálnych lokalít zameraným na distribúciu Android aplikácií: *www.appsapk.com*, *www.apkmaniafull.com* a *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 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*[1].

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

# 5 Analýza APK súborov

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 a reverzného inžinierstva APK súborov v tejto práci je získanie metadát o APK súbore, ktoré sú využívané v kapitole 6 a 7.

## 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 nástroje obsiahnuté v *Android Software Development Kit (SDK).Android SDK* je kolekcia štandardných nástrojov používaných pri vývoji a zostavení Android aplikácií.

### 5.1.1 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. Takéto súbory obsahujú nízkoúrovňový zdrojový kód. *ApkTool* podporuje debugovanie smali kódu [**apkTool**].

### 5.1.2 Dex2Jar

Nástroj podporujúci dekódovanie DEX súborov do formátu skompilovaných CLASS súborov .Výsledné CLASS súbory môžu byť prevedené do čitateľného kódu v jazyku Java pomocou dekompilátoru *JD-GUI*. Pracuje výhradne so súborom *classes.dex* a nepodporuje prevod binárnych XML do čitateľnej podoby.

### 5.1.3 AAPT

*Android Asset Packaging Tool* (*AAPT*) je štandardný nástroj obsiahnutý v *Android SDK*. Nástroj *AAPT* umožňuje vytvorenie, aktualizovanie a prezeranie súborov vo formáte APK. Dokáže skompilovať zdrojové súbory do binárnej formy a umožňuje aj ich čiastočnú dekompiláciu[**aapt**].

### 5.1.4 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.

## 5.2 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 veľkosti 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 5.1.4), v prípade zlyhania konverzie sa použije nástroj *ApkTool* (viď 5.1.1). Dáta získané analýzou tohto súboru zahŕňajú napríklad verziu aplikácie, použité prístupové oprávnenia alebo komponenty z ktorých sa aplikácia skladá

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

- Informácie o zdrojových súboroch[1] – informácie o zdrojoch aplikácie, 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 A.1.

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 vytvorená 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 o 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 popísanou v kapitole 5. 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

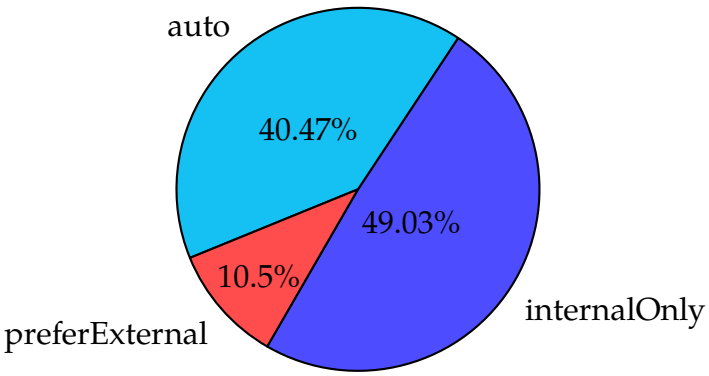
Analýzou databázy APK súborov sa zistilo, že medián veľkosti APK súborov je 5,26 MB, aritmetický priemer dosahuje hodnotu 10.19 MB. Aritmetický priemer je štatistický ukazovateľ, ktorý je citlivý na extrémne hodnoty. Keďže je jeho hodnota v porovnaní s mediánom takmer dvojnásobná, v našej databáze sa nachádzajú aplikácie, ktorých veľkosť výrazne prevyšuje najčastajšie veľkosti APK súborov.

### Počet súborov v APK balíku

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

### Inštalačná politika

Android poskytuje aplikáciám možnosť špecifikácie preferovaného pamäťového priestoru (interná alebo externá pamäť) a prípadnú možnosť presunutia nainštalovanej aplikácie (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 inštaláciu na externé pamäťové médium. Rozdelenie hodnôt je zobrazené v grafe na obrázku 6.1.




Obr. 6.1: Hodnoty atribútu *android:installLocation*

### Prístupové oprávnenia

Aplikácie najčastejšie deklarujú využívanie 4 prístupových oprávnení (viď 3.7.2). Medián počtu vyžadovaných oprávnení je 8. 10 najčastejšie využívaných oprávnení spolu s ich percentuálnym zastúpením v analyzovanej vzorke aplikácií je uvedených v tabuľke 6.1.

| 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 6.1: Najpoužívanejšie prístupové oprávnenia

### Komponenty aplikácií

Základnou funkčnou jednotkou aplikácie je aktivita (viď 3.7.7).Medián počtu aktivít medzi analyzovanými aplikáciami je 10, priemer dosahuje hodnotu 20,23. Aplikácie obsahujú najčastejšie 2 aktivity. Priemerný počet služieb definovaných v aplikácii je 3,99, medián je 1, no najčastejším prípadom je, že aplikácia nedefinuje žiadnu službu.

### Obrázkové súbory

Analýza ukázala, že aplikácie využívajú množstvo obrázkových súborov. Medián celkového počtu obrázkových súborov v APK balíčku je 210, aritmetický priemer dosahuje hodnotu 462 a najčastejším počtom obrázkových súborov je 5. Medián počtu rozdielnych obrázkov (veľkosť a rozlíšenie sa neberie do úvahy) je 134. Najčastejším formátom je PNG, aplikácia obsahuje priemerne 343 takýchto súborov.

### Využité vlastnosti

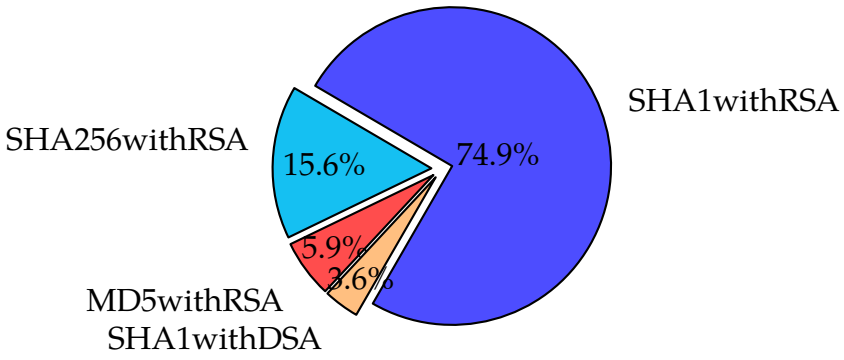
Aplikácie deklarujú nízky počet využívaných vlastností (viď 3.7.2). Aritmetický priemer je 1,44, medián a a modus sú nulové. Najčastejšie deklarované je využívanie vlastností uvedených v tabuľke 6.2.

| 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 6.2: Najpoužívanejšie vlastnosti

### 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. Percentuálne rozdelenie algoritmov použitých na podpis APK súborov je znázornený na obrázku 6.1.




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. 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 %. Najčastejšie lokalizácie aplikácií sú uvedené v tabuľke 6.3.

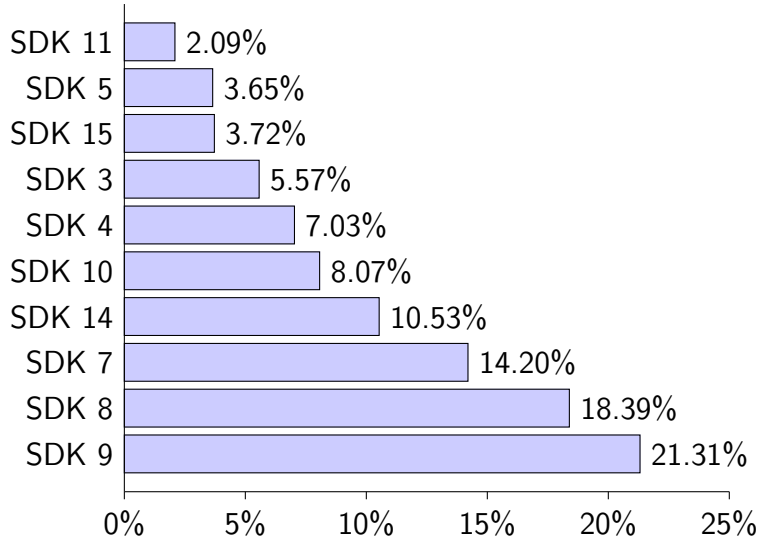
| Kód | Jazyk | % |
|---|---|---|
| es | španielsky | 61,7 |
| de | nemecký | 59,6 |
| fr | francúzsky | 59,4 |
| ru | ruský | 58,1 |
| ja | japonský | 57,6 |
| it | taliansky | 57,4 |
| ko | korejský | 56,9 |
| zh-rcn | čínsky (zjednodušený) | 55,6 |
| zh-rtw | čínsky (tradičný) | 54,0 |
| pt | portugalský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

Tabuľka 6.3: Lokalizácia aplikácií

### Verzie Android SDK

Najčastejšou najnižšou vyžadovanou verziou Android SDK je verzia 9 s 21,3% zastúpením. Táto verzia je asociovaná so systémom Android 2.3 až Android 2.3.2 *Gingerbread* a bola vydaná v novembri 2010. Nanižšie vyžadované verzie Android SDK v našej databáze APK súborov sú zobrazené v grafe 6.4 a graf na obrázku 6.1.

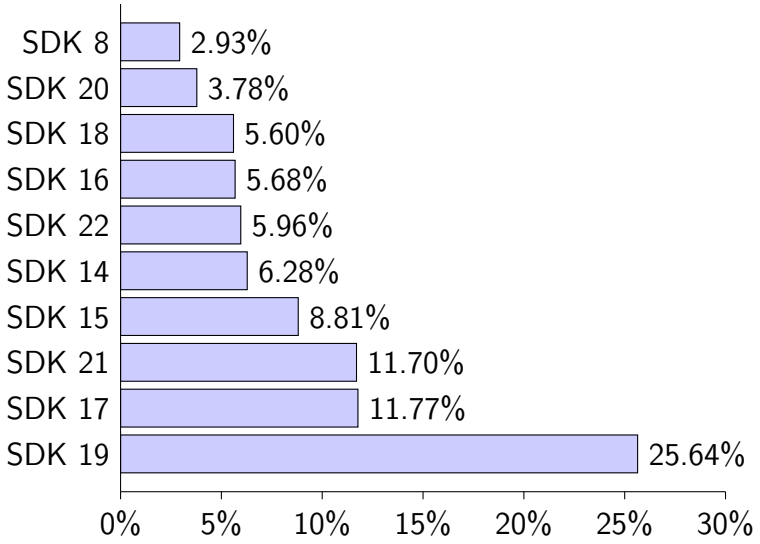
Až 25,64 % aplikácií je primárne určených na SDK verziu 19. SDK 19 bol vydaný koncom roka 2013 spolu s verziou Android 4.4 *KitKat*. Najčastejšie hodnoty primárnej verzie Android SDK obsahuje tabuľka 6.5 a graf na obrázku 6.1.




Obr. 6.3: Hodnoty atribútu *android:minSdk*

| **SDK** | **Android** | **Vydanie** | **%** |
|---|---|---|---|
| 9 | Android 2.3 Gingerbread | November 2010 | 21,3 |
| 8 | Android 2.2.x Froyo | Jún 2010 | 18,4 |
| 7 | Android 2.1.x Eclair | Január 2010 | 14,2 |
| 14 | Android 4.0 Ice Cream Sandwich | Október 2011 | 10,5 |
| 10 | Android 2.3.3 Gingerbread | Február 2011 | 8,1 |
| 4 | Android 1.6 Donut | September 2009 | 7,0 |
| 3 | Android 1.5 Cupcake | Máj 2009 | 5,6 |
| 15 | Android 4.0.3 Ice Cream Sandwich | December 2011 | 3,7 |
| 5 | Android 2.0 Eclair | November 2009 | 3,7 |
| 11 | Android 3.0.x Honeycomb | Február 2011 | 2,1 |

Tabuľka 6.4: Hodnoty najnižsej vyžadovanej verzie Android SDK




Obr. 6.4: Hodnoty atribútu *android:targetSdk*

| **SDK** | **Android** | **Vydanie** | **%** |
|---|---|---|---|
| 19 | Android 4.4 KitKat | Október 2013 | 25,6 |
| 17 | Android 4.2 – 4.2.2 Jelly Bean | November 2012 | 11,8 |
| 21 | Android 5.0 Lollipop | November 2014 | 11,7 |
| 15 | Android 4.0.3 Ice Cream Sandwich | December 2011 | 6,8 |
| 14 | Android 4.0 Ice Cream Sandwich | Október 2011 | 6,3 |
| 22 | Android 5.1 Lollipop | Marec 2015 | 6,0 |
| 16 | Android 4.1 – 4.1.1 Jelly Bean | Jún 2012 | 5,7 |
| 18 | Android 4.3 Jelly Bean | Júl 2013 | 5,6 |
| 20 | Android 4.4W Kitkat Watch | Jún 2014 | 3,8 |
| 8 | Android 2.2.x Froyo | Jún 2010 | 2,9 |

Tabuľka 6.5: Hodnoty cieľovej verzie Android SDK

# 7 Prebalené APK súbory

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 ktorým 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. Prebalovanie APK súborov má negatívny vplyv na vývojárov originálnej aplikácie. V prípade možnosti nákupu priamo z aplikácie[1], môžu byť výnosy presmerované z účtov originálnych vývojárov na účet ľudí, ktorí aplikáciu modifikovali. Ovplyvnené sú aj obchody 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 **[Zhauniarovich2014]**.

## 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 balíčkov **[buildingAndRunning]** umožňuje jednoduchú dekompiláciu.

V prípade modifikácie zdrojového kódu je možné použiť existujúce nástroje reverzného inžinierstva, napríklad *dex2Jar* alebo *ApkTool*, ktoré sú bližšie popísané v kapitole 5.1.

Modifikácia súboru *AndroidManifest.xml* je takisto možná. Pomocou existujúcich nástrojov je možné previesť ho do čitateľného XML súboru, ktorý je možné editovať a následne previesť späť do binár-

1. angl. in-app purchase

neho 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 (viď 3.1). V prípade detekcii narušenia integrity, Android zakáže inštaláciu 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 modifikácii 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 metódy analyzujúce 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 3.7.2). Riešenia sú založené na použití statickej analýzy kódu, dynamickej analýzy alebo na detekcií známych vzoriek škodlivého kódu[3] [**Huang2013**, **Chen2015**, **Milanova2005**, **Levchenko2011**, **Hanna2013**, **Zhou2012**, **Potharaju2012**].

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

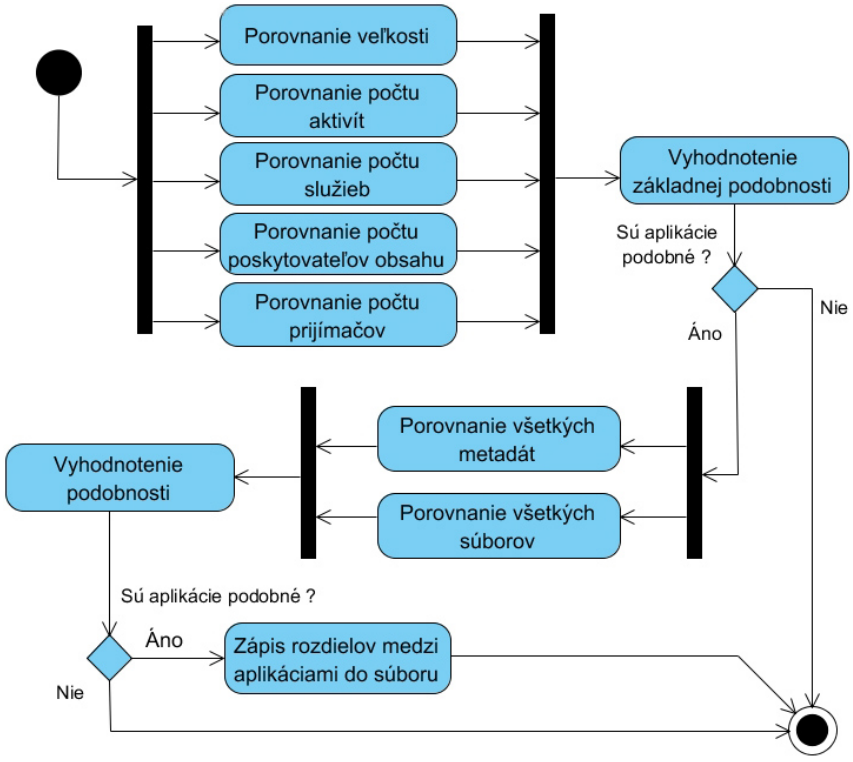
### 7.2.2 Detekcia pomocou podobnosti súborov

Prebalené APK súbory je možné úspešne detekovať prostredníctvom zhody súborov obsiahnutých v APK balíčkoch. Tento prístup využíva skutočnosť, že aplikácia nie je definovaná iba svojim zdrojovým kódom a funkcionalitou, ale je tvorená aj ďalšími dôležitými prvkami ako 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 balíčkov obsiahnutá aj v modifikovaných balíčkoch. Originálna aplikácia Opera Mini a verzia tejto aplikácie obsahujúca malvér, sa zhodujú v 230 z 234 súborov nachádzajúcich sa v príslušných APK balíčkoch [**Zhauniarovich2014**]. Riešenie prezentované v práci *FSquaDRA* [**Zhauniarovich2014**] 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 [**Zhauniarovich2014**].

## 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* [**Zhauniarovich2014**].

Prístup je založený na podobnosti súborov. Celková podobnosť aplikácií je určená na základe 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 kombinovanú s metadátami a informáciami o daných APK súboroch. 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.




Obr. 7.1: Postup párového porovnávania APK súborov

### 7.3.1 Implementácia

Funkcionalita porovnávania a detekcie modifikovaných APK balíčkov je implementovaná v programe *ApkAnalyzer*. Používateľ môže aplikáciu spúšťať a zadávať jej 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, ktoré sú vytvorené aplikáciou *ApkAnalyzer* počas analýzy APK balíčkov a obsahujú metadata o aplikáciách (viď kapitola 5). 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 metadáta 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.

Všetky tieto hodnoty sú číselné. Je nutné aby implementácia ich porovnávania bola funkčná nezávisle na veľkosti týchto číselných hodnôt. Taktiež je nutné zabezpečiť komutatívnosť, ktorá zaručí, že nezáleží na poradí porovnávania aplikácií a teda pre funkciu porovnávania $func$ platí

$$func(A, B) = func(B, A)$$

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*, na porovnanie sú využité SHA1 hashe súborov uložené v *MANIFEST.MF*. Separátne sú porovnávané súbory *classes.dex*, *resources.arsc*. Ostatné súbory sú porovnávané v rámci kategórií: obrázkové súbory, súbory definujúce vzhľad obrazoviek, všetky ostatné súbory a všetky súbory v APK balíčku.

Zhoda súborov medzi dvomi APK balíčkami je určená pomocou *Jaccard indexu*. Nech $A$ sú súbory v danej kategórií v jednom APK balíčku, $B$ sú súbory v danej kategórií v druhom porovnávanom APK balíčku.

$$JaccardIndex(A, B) = \frac{A \cap B}{A \cup B}$$

Aplikácie sú považované za podobné v prípade, že hodnota *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. Pri zhode 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 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 s 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.

### 7.3.2 Výsledky

Vyhľadávanie možných prebalených APK súborov pomocou navrhnutej metódy určilo v našej databáze viacero podozrivých aplikácií. Porovnávanie všetkých aplikácií v našej databáze znamenalo viac ako dva milióny vykonaných párových porovnaní. Celkový čas potrebný na výpočet dosiahol približne 40 hodín. Porovnanie bolo vykonané na výkonnom ale bežnom komoditnom harvéri[4]. V čase potrebnom na porovnanie nie je započítaný čas na analýzu APK súborov a vytvorenie JSON súborov obsahujúcich metadáta. Vzhľadom na objem zbieraných dát a dekompiláciu APK balíčku pomocou nástroja *ApkTool* počas analýzy (viď kapitola 5), je celkový čas potrebný na určenie prebalených súborov vyšší ako v pripade metódy *FSquaDRA*, avšak výstup našeho porovnania obsahuje informácie o rozdieloch medzi jednotlivými APK súbormi. Kritéria pre považovanie dvoch APK súborov za nadmieru podobné boli počas porovnávania nastavené na minimálnu zhodu 50 %, respektíve hodnotu *Jaccard indexu* 0,5. Veľké množstvo aplikácií bolo vyhodnotených ako podobné, lebo sa jednalo o rovnaké aplikácie v rozdielnych verziách. Takéto aplikácie je možné vyfiltrovať s využitím kategórie podobnosti, ktorá je každej porovnávanej dvojici priradená. *ApkAnalyzer* automaticky rozdelí výstupné súbory do podpriečinkov podľa typu podobnosti APK súborov. Bolo identifikovaných 161 dvojíc aplikácií, ktoré splňovali kritéria zhody, boli v rovnakej verzii, ale boli podpísané rôznymi certifikátmi.

4. Procesor Intel Core i7-4900MQ @ 2.80GHz , 16GB RAM, SSD disk

# 8 Záver

Cieľom práce bolo vytvorenie rozsiahlej databázy inštalačných APK súborov pre operačný systém Android. Získané APK súbory mali byť analyzované za účelom získania metadát, nad ktorými mala byť vykonaná analýza štatistických vlastností APK súborov z vytvorenej databázy. Ďalším cieľom bol návrh a implementácia jednoduchej metódy využívajúcej informácie o APK súboroch na detekciu potenciálne škodlivých prebalených aplikácií. Všetky ciele vytýčené zadaním práce sa podarilo splniť.

V rámci práce bola vytvorená databáza obsahujúca viac ako 20000 APK súborov. Získanie veľkého počtu APK súborov bolo možné vďaka automatizovaniu ich sťahovania prostredníctvom vyvinutého nástroja *ApkDownloader*. Časť aplikácií bola získaná z archívu vytvoreného v práci *Playdrone*, ktorý obsahuje aplikácie prevzaté z oficiálneho obchodu *Google Play* [**Viennot2014**]. Väčšina aplikácií však pochádza z neoficiálnych zdrojov, čím sa zvyšuje pravdepodnosť prítomnosti škodlivého softvéru v aplikáciách.

S využitím nástrojov reverzného inžinierstva boli APK súbory z našej databázi analyzované. Práca obsahuje detailný popis štruktúry a obsahu APK balíčkov so zameraním na súbory a atribúty využívané pri ich analýze. Bol implementovaný program *ApkAnalyzer*, ktorý poskytuje funkcionalitu na analýzu veľkého počtu APK súborov. Výstupom analýzy jedného APK balíčka je súbor obsahujúci metadáta o danom balíčku.

Na základe informácií o jednotlivých aplikáciách boli určené štatistické dáta nad množinou APK súborov. Vytvorená databáza poskytovala dostatočne veľký počet rozličných APK súborov na výpočet štatistických informácií.

Práca obsahuje návrh a implementáciu jednoduchej metódy detekcie prebalených APK balíčkov. Metóda je založená na zhode metadát a súborov obsiahnutých v APK súboroch. Túto funkcionalitu poskytuje aplikácia *ApkAnalyzer*, ktorá umožňuje detekovať dvojice nadmieru podobných APK súborov a zobraziť rozdiely medzi nimi. Pomocou navrhnutého algoritmu bolo detekovaných 161 dvojíc aplikácií, pri ktorých je veľká pravdepodbnosť modifikácie.

Programy *ApkAnalyzer* a *ApkDownloader* sú open source, môžu byť upravené a použité v ďalších prácach zaoberajúcich sa danou problematikou.

# A Prílohy

| **Atribút** | **Popis** |
|---|---|
| fileName | Názov analyzovaného APK súboru |
| sourceOfFile | Zdroj súboru |
| fileSize | Veľkosť APK súboru v bajtoch |
| dexSize | Veľkosť súboru *classes.dex* v bajtoch |
| arscSize | Veľkosť súboru *arscSize.dex* v bajtoch |
| packageName | Hodnota atribútu *package* v elemente *manifest* |
| versionCode | Hodnota atribútu *android:versionCode* v elemente *manifest* |
| installLocation | Hodnota atribútu *android:installLocation* v elemente *manifest* |
| numberOfActivities | Počet aktivít definovaných aplikáciou |
| numberOfServices | Počet služieb definovaných aplikáciou |
| numberOfContentProviders | Počet poskytovateľov obsahu definovaných aplikáciou |
| numberOfBroadcastReceivers | Počet komponent typu *BroadcastReceiver* definovaných aplikáciou |
| namesOfActivities | Názvy aktivít definovaných aplikáciou |
| namesOfServices | Názvy služieb definovaných aplikáciou |
| namesOfContentProviders | Názvy poskytovateľov obsahu definovaných aplikáciou |
| namesOfBroadcastReceivers | Názvy komponent typu *BroadcastReceiver* definovaných aplikáciou |
| usesPermissions | Názvy povolení využívaných aplikáciou |

| **Atribút** | **Popis** |
|---|---|
| usesLibrary | Názvy knižníc využívaných aplikáciou |
| permissions | Názvy povolení definovaných aplikáciou |
| permissionsProtectionLevel | Level ochrany povolení definovaných aplikáciou |
| usesFeature | Názvy vlastností využívaných aplikáciou |
| usesMinSdkVersion | Hodnota atibútu *android:minSdkVersion* elementu *uses-sdk* |
| usesTargetSdkVersion | Hodnota atribútu *android:targetSdkVersion* elementu *uses-sdk* |
| usesMaxSdkVersion | Hodnota atribútu *android:maxSdkVersion* elementu *uses-sdk* |
| supportsScreensResizeable | Hodnota atribútu *android:resizeable* elementu *supports-screens* |
| supportsScreensSmall | Hodnota atribútu *android:smallScreens* elementu *supports-screens* |
| supportsScreensNormal | Hodnota atribútu *android:normalScreens* elementu *supports-screens* |
| supportsScreensLarge | Hodnota atribútu *android:largeScreens* elementu *supports-screens* |
| supportsScreensXlarge | Hodnota atribútu *android:xlargeScreens* elementu *supports-screens* |
| supportsScreensAnyDensity | Hodnota atribútu *android:anyDensity* elementu *supports-screens* |

| **Atribút** | **Popis** |
|---|---|
| fileName | Názov súboru s certifikátom |
| signAlgorithm | Algoritmus použitý na podpis |
| signAlgorithmOID | OID algoritmu použitého na podpis |
| startDate | začiatok platnosti certifikátu |
| endDate | koniec platnosti certifikátu |
| publicKeyMd5 | MD5 hash verejného klúča |
| certBase64Md5 | Base64 MD5 hash certifikátu |
| certMd5 | MD5 hash certifikátu |
| version | Verzia cetifikátu |
| issuerName | Názov vydávateľa vo formáte definovanom RFC 2253 |
| subjectName | Názov subjektu vo formáte definovanom RFC 2253 |
| locale | Lokalizácie súboru *string.xml* |
| numberOfStringResource | Počet záznamov v súbore *string.xml* |
| pngDrawables | Počet PNG obrázkov |
| ninePatchDrawables | Počet 9.PNG obrázkov |
| jpgDrawables | Počet JPG obrázkov |
| gifDrawables | Počet GIF obrázkov |
| xmlDrawablesr | Počet XML obrázkov |
| ldpiDrawables | Počet obrázkov v ldpi priečinku |
| mdpiDrawables | Počet obrázkov v mdpi priečinku |
| hdpiDrawables | Počet obrázkov v hdpi priečinku |
| xhdpiDrawables | Počet obrázkov v xhdpi priečinku |
| xxhdpiDrawables | Počet obrázkov v xxhdpi priečinku |
| xxxhdpiDrawablesr | Počet obrázkov v xxxhdpi priečinku |
| tvdpiDrawables | Počet obrázkov v tvdpi priečinku |
| nodpiDrawables | Počet obrázkov v nodpi priečinku |

| **Atribút** | **Popis** |
|---|---|
| unspecifiedDpiDrawables | Počet obrázkov nezaradených v *dpi priečinku |
| rawResources | Počet súborov v *raw/* priečinku |
| layouts | Počet súborov v *res/layout** priečinkoch |
| differentLayouts | Počet rôznych súborov v *res/layout** priečinkoch |
| menu | Počet súborov v *res/menu* priečinku |
| dexHash | Hash súboru *classes.dex* prevzatý zo súboru *MANIFEST.MF* |
| arscHash | Hash súboru *arscHash.dex* prevzatý zo súboru *MANIFEST.MF* |
| drawableHash | Hashe a cesty k súborov z prečinku *res/drawable* prevzaté zo súboru *MANIFEST.MF* |
| layoutHash | Hashe a cesty k súborov z prečinku *res/layout* prevzaté zo súboru *MANIFEST.MF* |
| otherHash | Hashe a cesty k všetkým ostatným súborov v APK balíčku prevzaté zo súboru *MANIFEST.MF* |

Tabuľka A.1: Zbierané metadata o APK súbore