

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.

Podakovanie

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.

Klíčové slová

APK sůbor, Android, Apktool, malvér, analýza aplikací, 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	7
3	APK súbory	9
3.1	Priečink METAINF	9
3.2	Priečink res	11
3.3	Priečink lib	13
3.4	Priečink 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	17
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	22
3.7.11	Element uses-library	22
4	Databáza inštalačných APK súborov	25
4.1	Implementácia	26
5	Analýza APK súborov	27
5.1	Nástroje reverzného inžinierstva	27

5.1.1	ApkTool	27
5.1.2	Dex2Jar	27
5.1.3	AXML	28
5.1.4	AAPT	28
5.2	<i>Implementácia analýzy</i>	28
6	Štatistiky	31
6.1	<i>Získané dáta</i>	31
6.1.1	Podpis APK balíčka	34
7	Prebalené APK súbory	37
7.1	<i>Modifikácia APK súborov</i>	37
7.2	<i>Známe metódy detekcie prebalených APK súborov</i>	38
7.2.1	Detekcia pomocou analýzy zdrojového kódu	38
7.2.2	Detekcia pomocou podobnosti súborov	39
7.3	<i>Navrhnutá metóda detekcie prebalených APK súborov</i>	39
7.3.1	Implementácia	41
7.3.2	Výsledky	43
8	Záver	45
	Register	45
A	Prílohy	47

Zoznam tabuliek

4.1	Zdroje prevzatých APK súborov	25
6.1	Najpoužívanéjšie prístupové oprávnenia	33
A.1	Lokalizácia aplikácií	47
A.2	Najpoužívanéjšie vlastnosti	48
A.3	Hodnoty najnižšej vyžadovanej verzie Android SDK	48
A.4	Hodnoty cieľovej verzie Android SDK	49
A.5	Zbierané metadata o APK súbore	53

Zoznam obrázkov

2.1	Vrstevnatá architektúra systému Android	4
3.1	Typická štruktúra APK súboru	10
6.1	Hodnoty atribútu <i>android:installLocation</i>	32
6.2	Algoritmus podpisu APK balíčku	34
7.1	Postup párového porovnávania APK súborov	40

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 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 Google¹. 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

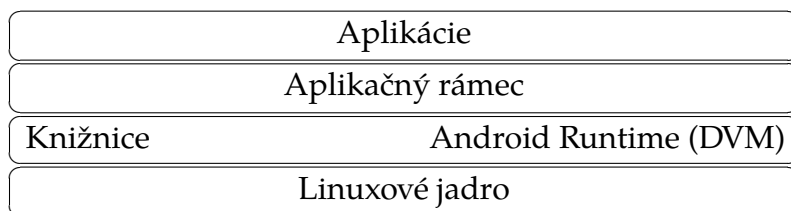
1. Google services

2. OPERAČNÝ SYSTÉM ANDROID

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 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 cukroví.

2.2 Architektúra systému

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



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*² používaného

2. Java Virtual Machine

pri jazyku Java. Virtuálny stroj *Dalvik* využíva nízkoúrovňovú funkcionálnosť 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 [**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 sprostredkujú prístup pomocou jazyka 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* [**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 aplikáciám zobrazovať upozornenia

2. OPERAČNÝ SYSTÉM ANDROID

- *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*³. Aplikácie môžu byť distribuované pomocou alternatívnych obchodov ako napríklad *Amazon Appstore*⁴ alebo *SlideMe*⁵. 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 aplikácie. Viacej informácií o prebalených aplikáciách sa nachádza v kapitole 7.

3. <https://play.google.com/store>

4. <http://www.amazon.com/mobile-apps>

5. <http://slideme.org/>

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 [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 prídanie 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

2. OPERAČNÝ SYSTÉM ANDROID

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 príslušný MIME typom *application/vnd.android.package-archive* [Freed2016].

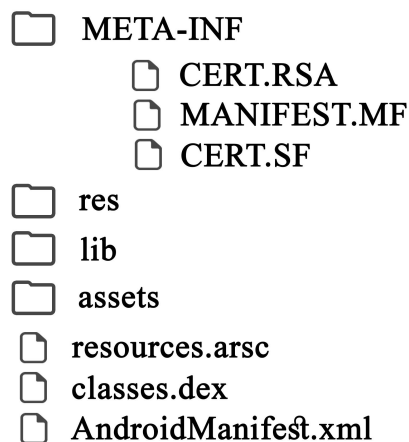
Štruktúra APK balíčkov vychádza z JAR¹ balíčkov – súborov používaných na distribúciu aplikácií alebo knižníc na platforme Java. Formát APK rozširuje všeobecnejší JAR formát o súbory, ktoré sú špecifické pre cieľovú platformu, ktorou je operačný systém Android. Zároveň si však ponecháva vlastnosti JAR súborov. APK balíčky sú archívne súbory v ZIP² formáte. Keďže APK používajú ZIP formát, k ich obsahu môžeme jednoducho prísť 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 obsahujúci 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.

1. Java archive

2. [https://en.wikipedia.org/wiki/Zip_\(file_format\)](https://en.wikipedia.org/wiki/Zip_(file_format))



Obr. 3.1: Typická štruktúra APK súboru

CERT.RSA

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

MANIFEST.MF

Súbor obsahujúci relatívne cesty a SHA-1 hashe³ 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: Vvga01jpW3iS1nBBikD/urdbN58=

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

3. 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: Vvga01jpW3iS1nBBikD/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 priečinky [providingRes]

- animator - obsahuje XML súbory definujúce property animácie⁴
- anim - obsahuje XML súbory definujúce tween animácie⁵, 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 rozmiestnenie prvkov na obrazovke
- menu - obsahuje XML súbory definujúce menu aplikácie
- raw - obsahuje súbory, ktoré musia byť uložené a použité v nekomprimovanej forme a kvalite
- values - obsahuje súbory vo formáte XML definujúce hodnoty textových refazcov, 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 prispôbenie 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

4. Animácie definované pomocou zmeny atribútov vykreslovaných objektov

5. 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 ISO 639-1⁶ 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 – 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.

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

3. APK SÚBORY

Na rozdiel od priečinku *res/raw/*, zdrojovým súborom v umiestnenom 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 funkcionality 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 nástroja *dx* je jediný súbor obsahujúci skompilovaný celý výkonný zdrojový kód aplikácie – *classes.dex*. Tento súbor je skompilovanou 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 aplikáciu

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.

```
<?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"
```

3. APK SÚBORY

```
        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 užívateľovi. Pre systém Android neposkytuje informáciu o verzii aplikácie, 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 častiam kódu je z dôvodu ochrany limitovaný. Android využíva princíp povolení⁷. 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

7. angl. permissions

3. APK SÚBORY

používané aplikáciami, ktoré vyžadujú prístup k chránenej časti poskytovanej danou aplikáciou.

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 aplikácií 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 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 vlastnosť⁸ 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.

8. angl. feature

3. APK SÚBORY

Element obsahuje nasledujúce atribúty [**elUsesFeature**]:

- *android:name* – názov vyžadovanej vlastnosti
- *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 prispôbiť rôznym veľkostiam obrazoviek
- *android:anyDensity* – indikuje, že aplikácia obsahuje zdrojové súbory vhodné pre obrazovky s rôznou hustotou 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 jednot-

livé prípady použitia aplikácie, implementujú časť grafického užívateľského rozhrania a zabezpečujú komunikáciu s užívateľom. Tieto triedy sú časťou 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 služba⁹. 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 funkcionality 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 štruktúrovaného prístupu k dátam spravovaným aplikáciou. Poskytovatelia obsahu¹⁰ 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* [**elContentProvider**]. Systém Android nerozpoznáva poskytovateľov obsahu nedefinovaných v *AndroidManifest.xml*.

9. angl. service

10. angl. content providers

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 aktivity¹¹ aj 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 aplikácii 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.

11. Intent

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 dostupný pre zariadenie *Galaxy Nexus* s operátorom *T-Mobile* [Viennot2014]. Naša databáza obsahuje 8200 najstiahovanejších 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 neposkytuje grafické užívateľské rozhranie, ale užívateľ môže zadávať parametre prostredníctvom príkazového riadku. Podporuje sťahovanie aplikácií získaných pomocou projektu *Playdrone* alebo z neoficiálnych lokalít zameraným na distribúciu Android aplikácií *www.appsapk.com*, *www.apkmaniafull.com* alebo *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 funkcionality 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*².

2. <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 funkcionality 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 dekodovať zdroje aplikácie do takmer originálnej podoby. Do čitateľnej podoby prevádza súbory *resources.arsc*, *classes.dex* aj binárne XML súbory. Z dekodovaných súborov umožňuje opätovné zostavenie APK súboru. Súbor *classes.dex* je dekompilovaný do súborov vo formáte SMALI. Smali súbory obsahujú nízkoúrovňový kód na úrovni assembleru. ApkTool podporuje debugovanie smali kódu [**apkTool**].

5.1.2 Dex2Jar

Nástroj podporujúci dekodovanie 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 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

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 dekompiláciu[**aapt**].

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 prispôbená 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 aplikácii.

Zbierané metadáta je možné rozdeliť do piatich kategórií:

- Základné informácie o APK súbore – v tejto kategórii 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 5.1.3), 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 rozbaľiť
- Informácie o zdrojových súboroch¹ – informácie o zdrojoch aplikácie, napríklad formát alebo veľkosť obrázkových súborov, počet lokalizácií aplikácie alebo počet surových neskompromovaných 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 uchováame 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.5.

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

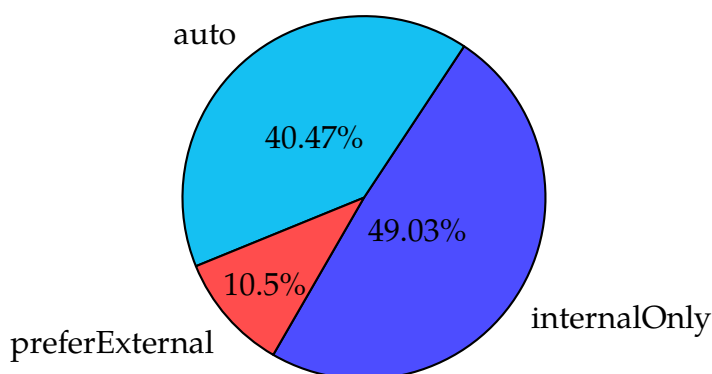
Analýzou databázy APK súborov sa zistilo, že stredná hodnota veľkosti APK súborov je 5,26 MB, priemer dosahuje hodnotu 10.19 MB.

Počet súborov v APK balíku

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

Inštalčná 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 6.1.



Obr. 6.1: Hodnoty atribútu *android: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. Nanižšie vyžadované verzie Android SDK v našej databáze APK súborov sú zobrazené v grafe A.3.

Až 25,64 % aplikácií je primárne určených na SDK verziu 19.

Prístupové oprávnenia

Aplikácie najčastejšie deklarujú využívanie 4 prístupových oprávnení (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 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

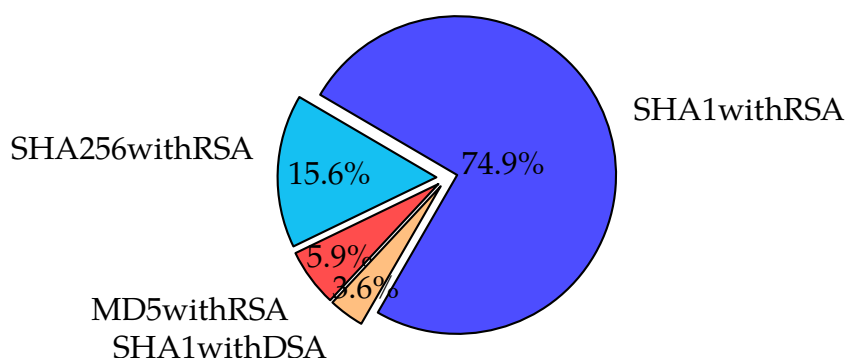
Tabuľka 6.1: Najpoužívanéjšie prístupové oprávnenia

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 tabuľke 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. Percentuálne rozdelenie algoritmov použitých na podpis APK súborov je znázornený na grafe 6.1.1.



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 A.1.

Obrázkové súbory

Analýza ukázala, že aplikácie využívajú množstvo obrázkových súborov. Stredná hodnota celkového počtu obrázkových súborov v APK balíčku 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 343 takýchto súborov.

7 Prebalené APK súbory

Pojem prebalený súbor označuje APK balíčky, ktoré boli modifikované, no navonok sa prezentujú ako originálne neupravené aplikácie. Časť tým prípadom je, že takéto aplikácie pochádzajú z oficiálneho zdroja Android aplikácií – *Google Play Store*, sú upravené a následne redistribuované pomocou neoficiálnych zdrojov. Takéto aplikácie si spravidla ponechávajú dizajn a funkcionality 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. Prebalené aplikácie majú negatívny vplyv na vývojárov originálnej aplikácie. V prípade, že aplikácia obsahuje možnosť nákupov priamo z aplikácie¹, 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 [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 ho previesť do čitateľného XML

1. angl. in-app purchase

súboru, ktorý je možné editovať a následne previesť späť do binárneho XML formátu. Túto funkcionality 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 detekcie 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 identifikuje². 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 detekcii známych vzoriek škodlivého kódu³ [Huang2013, Chen2015, Milanova2005, Levchenko2011, Hanna2013, Zhou2012, Potharaju2012].

2. angl. self-signed certificate

3. angl. signature based

7.2.2 Detekcia pomocou podobnosti súborov

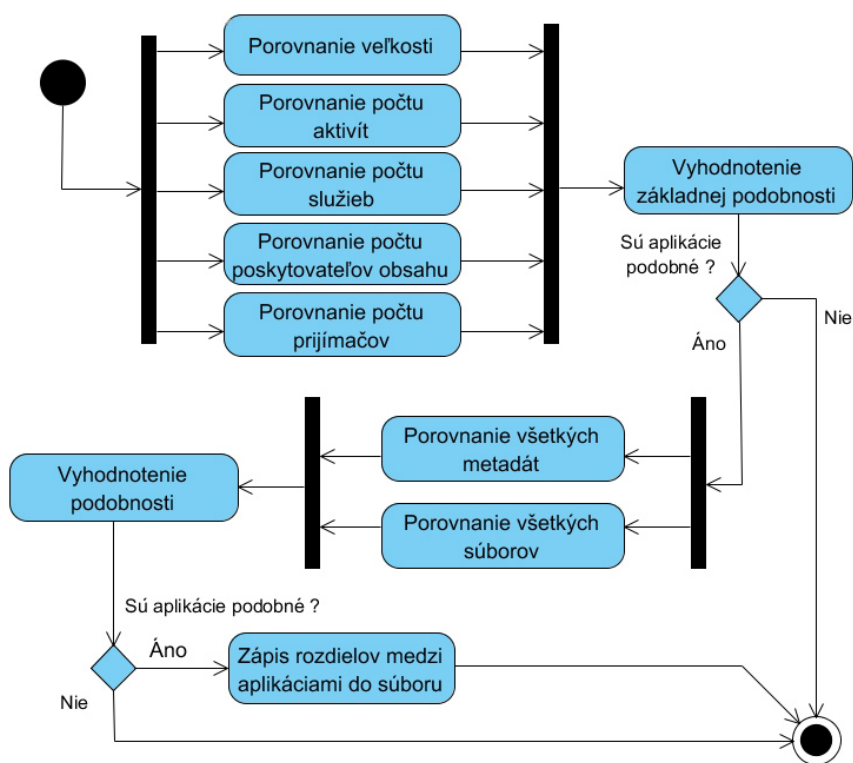
Prebalené APK súbory je možné úspešne detekovať prostredníctvom zhody súborov obsiahnutých v APK balíčkoch. Tento prístup využíva skutočnosť, že aplikácia nie je definovaná iba svojim zdrojovým kódom a funkcionalitou, ale je tvorená aj ďalšími dôležitými prvkami ako sú používateľské prostredie alebo 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 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* 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].

7. PREBALENÉ APK SÚBORY

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 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.

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 porovnávané 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 porovnávajú 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 a všetky ostatné súbory.

Zhoda súborov medzi dvomi APK balíčkami je určená pomocou *Jaccard indexu*. Nech A sú súbory v danej kategórii v jednom APK balíčku, B sú súbory v danej kategórii 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 porovnávajú 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 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.

7.3.2 Výsledky

Vyhľadávanie možných prebalených APK súborov pomocou navrhnutej metódy určilo v našej databáze viacero podozrivých aplikácií. Párové porovnávanie všetkých aplikácií v našej databáze znamenalo 2120770 vykonaných porovnaní. Celkový čas potrebný na výpočet dosiahol približne 40 hodín. Funkcionalita porovnávania bola vykonaná na výkonnom ale bežnom komoditnom harvéri. 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 prípade metódy *FSquaDRA*, avšak výstup nášeho porovnania obsahuje informácie o zmenách 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 porovnáwanej dvojici priradená. Bolo identifikovaných 161 dvojíc aplikácií, ktoré splňovali kritéria zhody a boli v rovnakej verzii, ale boli podpísané rôznymi certifikátmi.

8 Záver

TBD

Register

α , 31

dummy text, 31

T_EX, 31

vehicles

 speed cars, 31

 trucks, 31

A Prílohy

Kód	Jazyk	%
es	španielsky	61,7
de	nemecký	59,6
fr	francúzsky	59,4
ru	ruský	58,1
ja	japonský	57,6
it	talianky	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: Najpoužívanéjšie vlastnosti

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.3: Hodnoty najnižšej 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.4: Hodnoty cieľovej verzie Android SDK

A. PRÍLOHY

Atribút	Popis
fileName	Názov analyzovaného APK súboru
sourceOfFile	Zdroj súboru
fileSize	Veľkosť APK súboru v bajtoch
dexSize	Veľkosť súboru <i>classes.dex</i> v bajtoch
arscSize	Veľkosť súboru <i>arscSize.dex</i> v bajtoch
packageName	Hodnota atribútu <i>package</i> v elemente <i>manifest</i>
versionCode	Hodnota atribútu <i>android:versionCode</i> v elemente <i>manifest</i>
installLocation	Hodnota atribútu <i>android:installLocation</i> v elemente <i>manifest</i>
numberOfActivities	Počet aktivít definovaných aplikáciou
numberOfServices	Počet služieb definovaných aplikáciou
numberOfContentProviders	Počet poskytovateľov obsahu definovaných aplikáciou
numberOfBroadcastReceivers	Počet komponent typu <i>BroadcastReceiver</i> definovaných aplikáciou
namesOfActivities	Názvy aktivít definovaných aplikáciou
namesOfServices	Názvy služieb definovaných aplikáciou
namesOfContentProviders	Názvy poskytovateľov obsahu definovaných aplikáciou
namesOfBroadcastReceivers	Názvy komponent typu <i>BroadcastReceiver</i> 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 atribútu <i>android:minSdkVersion</i> elementu <i>uses-sdk</i>
usesTargetSdkVersion	Hodnota atribútu <i>android:targetSdkVersion</i> elementu <i>uses-sdk</i>
usesMaxSdkVersion	Hodnota atribútu <i>android:maxSdkVersion</i> elementu <i>uses-sdk</i>
supportsScreensResizeable	Hodnota atribútu <i>android:resizeable</i> elementu <i>supports-screens</i>
supportsScreensSmall	Hodnota atribútu <i>android:smallScreens</i> elementu <i>supports-screens</i>
supportsScreensNormal	Hodnota atribútu <i>android:normalScreens</i> elementu <i>supports-screens</i>
supportsScreensLarge	Hodnota atribútu <i>android:largeScreens</i> elementu <i>supports-screens</i>
supportsScreensXlarge	Hodnota atribútu <i>android:xlargeScreens</i> elementu <i>supports-screens</i>
supportsScreensAnyDensity	Hodnota atribútu <i>android:anyDensity</i> elementu <i>supports-screens</i>

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 kľúča
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 <i>string.xml</i>
numberOfStringResource	Počet záznamov v súbore <i>string.xml</i>
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 nezarađených v *dpi priečinku
rawResources	Počet súborov v <i>raw/</i> priečinku
layouts	Počet súborov v <i>res/layout*</i> priečinkoch
differentLayouts	Počet rôznych súborov v <i>res/layout*</i> priečinkoch
menu	Počet súborov v <i>res/menu</i> priečinku
dexHash	Hash súboru <i>classes.dex</i> prevzatý zo súboru <i>MANIFEST.MF</i>
arscHash	Hash súboru <i>arscHash.dex</i> prevzatý zo súboru <i>MANIFEST.MF</i>
drawableHash	Hashe a cesty k súborov z priečinku <i>res/drawable</i> prevzaté zo súboru <i>MANIFEST.MF</i>
layoutHash	Hashe a cesty k súborov z priečinku <i>res/layout</i> prevzaté zo súboru <i>MANIFEST.MF</i>
otherHash	Hashe a cesty k všetkým ostatným súborov v APK balíčku prevzaté zo súboru <i>MANIFEST.MF</i>

Tabuľka A.5: Zbierané metadata o APK súbore